Mechanisms of Tin Oxide Gas Sensor Response

Andrew Peter Lee B.App.Sc (Hons)

Submitted in fulfilment of the requirements

for the Degree of

Doctor of Philosophy

Chemistry, Launceston
University of Tasmania (April, 2001)

If we knew what it was we were doing, it would not be called research, would it?

Albert Einstein

Declaration

This thesis contains no material which has been accepted for a degree or diploma by the University or any other institution, except by way of background information and duly acknowledged in the thesis. To the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due acknowledgement is made in the text of the thesis.

Andrew P. Lee 29/4/2001

This thesis may be made available for loan and limited copying in accordance with the *Copyright Act 1968*.

Abstract

Tin oxide gas sensors are widely used for the detection of combustible gases in oxygen-rich atmospheres. Adsorbed oxygen species withdraw electron density from the surface of the SnO_2, increasing its electrical resistance. At elevated temperatures, around 400 °C, combustible analyte gases displace or react with adsorbed oxygen, increasing SnO_2 surface electron density and thus decreasing its electrical resistance. Sensor resistance has been found to vary non-linearly with combustible gas concentration in a manner that has not been satisfactorily explained despite thirty years of research into the sensing mechanism.

The operating temperature of tin oxide gas sensors is critical information in studies of their response mechanism, yet has seldom been reported accurately in the literature. In the current work, a new method of determining sensor temperature radiometrically has been developed and used to determine the surface temperature of two types of Figaro tin oxide sensors. The operating voltage-temperature relationships for these sensors were found to be pseudo-linear and are reported as $T = 103V+214\pm3$ K for the Figaro TGS813 sensor with its base removed, $T = 101V+224\pm5$ K for the TGS813 with its base attached, and $T = 106V+238\pm5$ K for the Figaro TGS2611 sensor. These results indicate that sensor temperatures are significantly higher than most previously reported estimates.

Investigations of TGS2611 sensor response reveal that oxygen exhibits nearly ideal (Langmuir) adsorption behaviour on these SnO_2-based gas sensors. An equation for the response of these devices to oxygen has been developed from a combination of accepted adsorption and electrical conduction theories. Fits of this equation to low and high sensor temperature oxygen response curves confirm

previous findings regarding the speciation of adsorbed oxygen, ie at temperatures below ~ 170 °C, oxygen adsorbs non-dissociatively, while above this temperature it adsorbs dissociatively. From the temperature-dependent response of a TGS2611 sensor operating in air, enthalpies of adsorption have been calculated for non-dissociative (ΔH = -35.4 kJ mol^{-1}) and dissociative (ΔH = -126.7 kJ mol^{-1}) oxygen adsorption. These values are characteristic of physisorption and chemisorption of oxygen to the surface respectively.

A combination of infrared studies and measurements of sensor resistance for a TGS2611 sensor have shown that *n*-alkanes adsorb competitively with oxygen onto the sensor surface. A competitive adsorption (Hinshelwood) mechanism is thus proposed for the response to combustible gases, using the previously developed oxygen response equation as a basis. The sets of equations representing this model are too difficult to solve implicitly, so their validity has been demonstrated using Monte Carlo-type computer simulations of sensor response to single *n*-alkanes and binary mixtures of these gases. Detailed information has been acquired about the adsorption and kinetic behaviour of oxygen and *n*-alkanes on tin oxide sensors, including the influence of alkane chain length on static and dynamic temperature responses.

The research in this thesis represents the first satisfactory explanation, in terms of heterogeneous adsorption and catalysis theory, of many aspects of sensor response, including the influence of oxygen on sensor resistance, the non-linear analyte response behaviour, the characteristic analyte resistance/temperature profiles, and the complex response of analyte mixtures.

Acknowledgements

Firstly, I would like to thank Dr. Brian Reedy for his detailed technical supervision, encouragement, enthusiasm and friendship.

Thanks to Associate Professor Don McWilliam for providing me with the opportunity to undertake this project, and thanks to Professor Peter Alexander for introducing me to the field.

I wish to acknowledge the Australian Research Council for the large grant (No. A29703193) from which the research funding and my scholarship were drawn.

To the staff and students of the School of Applied Science, thanks for the help, support and friendship over the last three years.

Thanks to Stephen Rowlings for his assistance, and friendship.

A big thanks to my family, Peter, Helen and Meg, without their support and encouragement, this project would never have been completed.

Finally, to Jodie, this work is dedicated to you. Your friendship, inspiration and assistance made it all possible.

Publications and prizes

A.P. Lee, B.J. Reedy, Temperature modulation in semiconductor gas sensing, *Sensors and Actuators B* **60** (1999) 35-42.

A.P. Lee, B.J. Reedy, Application of radiometric temperature determination methods to semiconductor gas sensors, *Sensors and Actuators B* **69** (2000) 37-45.

A.P. Lee, B.J. Reedy, P.W. Alexander, A new model for tin oxide gas sensor response, *First Prize: Poster presentation at RACI, R & D Topics Meeting*, Hobart, December (1998).

A.P. Lee, B.J. Reedy, P.W. Alexander, A new model for tin oxide gas sensor response, *First Prize: Poster presentation at Australian International Symposium on Analytical Science*, Melbourne, July (1999).

Table of Contents

Declaration i

Abstract ii

Acknowledgements iv

Publications and prizes v

Chapter 1 Introduction 1

1.1 Project Background 2

1.2 Overview of metal oxide gas sensors 3

1.2.1 Brief History of development 3

1.2.2 Sensing material-SnO_2 6

1.2.3 Principle of operation 7

1.2.4 General response characteristics 8

1.2.5 Temperature dependence of response 12

1.2.6 Dopants and promoters 14

1.3 Where to from here? 16

References 17

Chapter 2 Equipment 19

2.1 Introduction 20

2.2 Gas sensors 20

2.2.1 TGS813 21

2.2.2 TGS2611 24

2.3 FTIR 29

2.3.1 Spectrometer 29

2.3.2 IR gas cell 30

2.4 Gas mixing/dilution system 31

2.4.1 Design 31

2.4.2 Gases and analytes 34

2.4.3 Operation 35

2.5 Data acquisition and control system – hardware 36

2.5.1 Microprocessor 37

2.5.2 Digital to analogue (D/A) interface board 38
2.5.3 Signal conditioning board 41
2.6 Data acquisition and control system – software 45
2.6.1 Real time package 46
2.6.2 Logging package 47
2.6.3 Program listings 48
References 49

Chapter 3 Sensor temperature measurement 51
3.1 Introduction 52
3.2 Theory and methodology 55
3.3 Experimental 59
3.3.1 Sensors 59
3.3.2 Melting point observations 60
3.3.3 Collection of sensor emission spectra 61
3.3.4 Data analysis 62
3.3.5 Infrared thermometer (IRT) 62
3.3.6 Variation of heater resistance with temperature (TGS2611 sensor) 63
3.4 Results and discussion 63
3.4.1 Determination of $R(\overline{\nu})\varepsilon(\overline{\nu})$ 63
3.4.2 Temperature measurements and heat loss model 69
3.4.3 Calibration of temperature-resistance relationship in TGS26XX sensors 75
3.5 Conclusion 79
References 80

Chapter 4 Oxygen response studies 83
4.1 Introduction 84
4.2 Theory – Langmuir adsorption 89
4.3 Experimental 90
4.3.1 Sensor response versus oxygen pressure studies 90
4.3.2 Dynamic oxygen adsorption studies 91

4.3.3 Sensor response versus temperature studies 92
4.4 Results and discussion 93
4.4.1 Sensor response versus oxygen pressure studies 93
4.4.2 Dynamic oxygen adsorption studies 101
4.4.3 Sensor response versus temperature studies 102
4.5 Conclusion 106
References 107

Chapter 5 Investigation and simulation of sensor response to *n*-alkanes 110
5.1 Introduction 111
5.2 Theory 114
5.2.1 Competitive adsorption 114
5.2.2 Surface reactions – Rideal mechanism 117
5.2.3 Surface reactions – Hinshelwood mechanism 117
5.2.4 Monte Carlo simulation techniques 119
5.3 Experimental 120
5.3.1 Investigation of n-alkane effect on sensor resistance in absence of oxygen 120
5.3.2 FTIR monitoring of n-alkane oxidation 120
5.3.3 Collection of sensor response data for n-alkanes (static temperature) 121
5.3.4 Collection of sensor response data for n-alkanes (dynamic temperature) 121
5.3.5 Collection of sensor response data for binary n-alkane mixtures (static temperature) 122
5.3.6 Oxygen adsorption simulation 123
5.3.7 Hinshelwood simulation – single analyte 124
5.3.8 Hinshelwood simulation – two analytes 127
5.4 Results and discussion 127
5.4.1 Effect of n-alkane on sensor resistance in absence of oxygen 127
5.4.2 Alkane oxidation and competitive adsorption 128
5.4.3 Oxygen adsorption simulation 131

5.4.4 n-Alkane oxidation and Hinshelwood simulation – single analyte 133
5.4.5 n-Alkane oxidation and Hinshelwoos simulation – binary mixtures 139
5.5 Conclusion 142
References 144

Chapter 6 Concluding remarks 147
6.1 Achievements 148
6.2 Future work 150

Appendices 151
1. Real time package – Tattletale program 152
2. Real time package – Macintosh program 154
3. Logging package – Tattletale program 174
4. Oxygen adsorption simulation 176
5. Hinshelwood simulation – single analyte 179
6. Hinshelwood simulation – two analytes 183

Chapter 1

Introduction

1.1 Project background

Metal oxide gas sensors have been extensively used for the detection of combustible gases since their development about thirty years ago. In spite of a large amount of research and development into these sensors, two problems remain which will limit their further application in qualitative and quantitative gas analysis. The first problem involves the highly non-linear temperature-dependent responses these sensors exhibit to both individual combustible gases and mixtures of gases. To date, a satisfactory mechanism for these sensor responses has not been proposed. The second problem is the lack of a technique to reliably measure the surface temperature of these devices, without which research into the first problem will be hindered.

This thesis addresses both of these problems, documenting the findings of research into the surface science and response mechanisms of tin (IV) oxide (SnO_2) based gas sensors.

1.2 Overview of metal oxide gas sensors

1.2.1 Brief history of development

Semiconductor metal oxides sensitive to the presence of reducing gases have been extensively studied since the 1960s [1]. Although it had long been known that various gases affect the electrical conductivity of metal oxides, it was not until the work of Seiyama [2] and Taguchi [3] in 1962 that the idea of using metal oxides as combustible gas detectors emerged. Seiyama reported that combustible gases could be detected from the resistance change of a thin film of zinc oxide, while Taguchi reported that a heated, sintered block of SnO_2 could also detect combustible gases through a resistance change. In 1967, both Shaver and Loh [4,5] reported that 'doping' with noble metals could increase the sensitivity of SnO_2-based devices. Following this discovery, in 1968, Taguchi established a private enterprise, named Figaro Engineering Inc., to research and manufacture sensors based on metal oxides. Mass production and sales of 'TGS' sensors based on Taguchi's original patent [3] commenced in 1968. These sensors used sintered pellets of tin oxide directly heated by embedded heater coils (Figure 1.1).

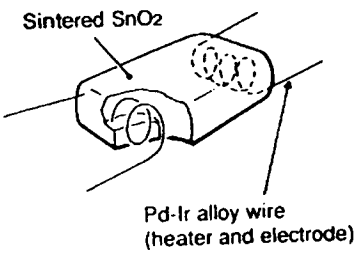


Figure 1.1 Directly heated sensor element as used in Figaro TGS1XX series sensors (from Figaro [6])

Taguchi continued development of sensor design, and was granted a United States patent for an indirectly heated sensor in 1972 [7]. This sensor consisted of a small alumina tube onto which gold electrodes were printed to facilitate resistance measurement of the sintered SnO_2 which covered the electrodes. A tightly wound heater coil placed inside the tube provided the necessary heating (Figure 1.2). This sensor design has become widely known as the Taguchi sensor and is the basic design of the Figaro TGS8XX series sensors. Until the release of a new sensor design in 1996, Figaro sold in excess of 80 million sensors based on the original designs [6].

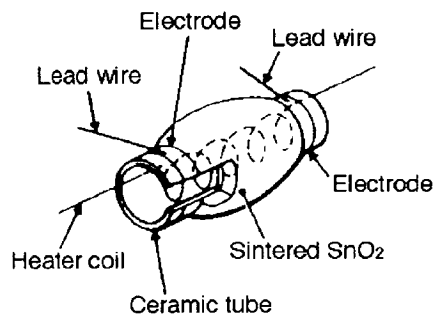


Figure 1.2 Taguchi or indirectly heated sensor element as used in Figaro TGS8XX series sensors (from Figaro [8])

The design of commercial sensors changed little until the 1990s, when a new company was founded in Britain. Capteur Sensors & Analysers Ltd claim to employ advanced ceramic fabrication techniques in their manufacturing process to produce thick-film mixed-metal oxide sensors, with enhanced performance and stability over SnO_2-based devices [9]. Figaro in response to Capteur's drive to capture market share released a new range of thick film sensors in 1996. The TGS2XXX series sensors also utilise thick-film printing technology, producing sensors with high levels of consistency and lower power consumption than

previous models [10]. Figaro claim the TGS2XXX series sensors use metal oxides other than SnO_2, but the TGS26XX series which were the first to be released and are the most widely used, still use SnO_2 as the basis of the sensing material. Although these thick-film devices improve on the indirectly-heated design of the Taguchi sensor, their design is more closely related to the earlier, directly heated devices. Figure 1.3 shows the sensing element of a typical thick film device with the heating element embedded in the substrate material.

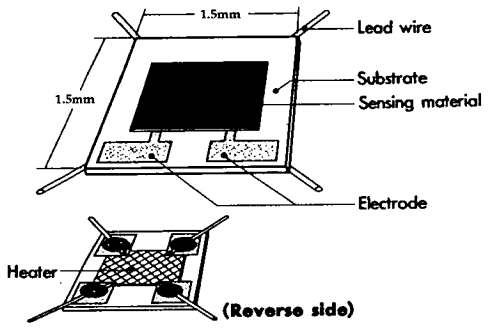


Figure 1.3 Typical thick film sensing element as used in Figaro TGS26XX series sensors (from Figaro [10])

All the metal oxide sensor designs discussed have advantages as combustible gas detectors over other technologies. These include:

- sensitivity (often sub ppm gas concentrations are detectable)
- low cost (typical Figaro TGS2XXX devices cost less than US $ 10)
- consistency (enabling simple sensor replacement)
- low power consumption

Their disadvantages include:

- non-linear response
- lack of selectivity
- susceptibility to poisoning by silicone, sulphur- and halogen-containing compounds

Although the latest trend in sensor design has been the investigation and use of metal oxides other than SnO_2, this material remains the most common, with the majority of installed gas sensors using it as the basis of their sensing material.

1.2.2 Sensing material-SnO_2

The majority of metal oxide gas sensors manufactured to date are constructed of SnO_2 with 'dopants' added to partially control response characteristics (see Section 1.2.6). Tin (IV) oxide (mineral form cassiterite) has the tetragonal rutile lattice structure of TiO_2 with a hexagonal close packed anion lattice where half the octahedral holes in the lattice are occupied by cations. This structure results in a high bulk electrical resistivity when stoichiometric. Although the bulk structure of SnO_2 is of interest, as for most other metal oxides, little is really known about its bulk properties [11]. On the other hand, much work has been done on the surface properties of SnO_2, and it is these that give rise to the gas-sensing ability of this material.

Preparation of all SnO_2 for gas sensor applications involves 'activation' by annealing at temperatures in excess of 700 °C [12]. It is widely accepted that this annealing process removes a mixture of bridging and in-plane oxygen anions from the surface (mostly the (110) surface, as this is the most thermodynamically stable [13]). The resulting surface of the activated material consists of tin cations

in various bonding arrangements and oxidation states, some of which have a delocalised electronic structure [13]. It is the adsorption of various gases onto this 'activated' surface, which alters the surface electronic properties of the SnO_2 and enables conductance measurements to be used for the detection of gases.

1.2.3 Principle of operation

It is widely accepted that the mechanism responsible for the conductance change of SnO_2 in the presence of combustible gases is the modulation of the concentration of adsorbed oxygen species on the SnO_2 surface [12-14]. Due to its electron affinity, adsorbed oxygen withdraws electron density from the SnO_2 surface, thus increasing the electrical resistance. Sensors typically operate at elevated temperatures, eg in the region of 400 °C. At these temperatures, the resistance of the sensor drops to readily measurable values (ambient temperature resistances are typically greater than 20 MΩ), due to the thermal desorption of oxygen. The introduction of a combustible gas into the atmosphere will result in a further decrease in sensor resistance due to oxidation of the reducing gas by adsorbed oxygen. This combustion removes adsorbed oxygen from the surface, releasing electron density back into the SnO_2, and thus decreasing the resistance. Figure 1.4 shows a diagram of the basic gas analyte gas detection mechanism of a SnO_2-based sensor. The magnitude of resistance variation will depend upon the analyte gas, the temperature of the sensor surface and the preparation of the sensing material itself, eg inclusion of 'dopants'.

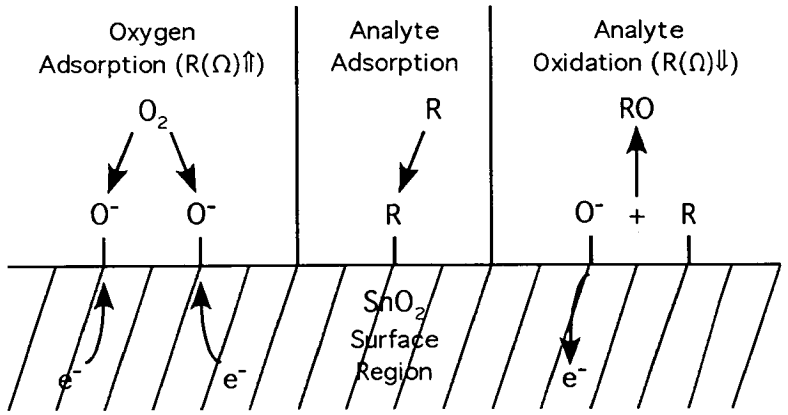


Figure 1.4 Gas detection mechanism in a SnO_2-based sensor

1.2.4 General response characteristics

Typical resistance versus concentration responses of SnO_2 gas sensors are highly non-linear, making quantitative measurements somewhat difficult. Most analyte gases give rise to 'power-law' responses of the form $R = kp^{-n}$ where p represents the pressure of the reducing gas, and k and n are constants. Although this equation resembles the Freundlich isotherm, these observed 'power-laws' are simply empirical fits, with the underlying mechanisms being far more complex (see Chapters 4 and 5). Figure 1.5 shows some typical non-linear responses of a TGS813 sensor to various analyte gases (the TGS813 is possibly the most widely used and studied Taguchi-style sensor).

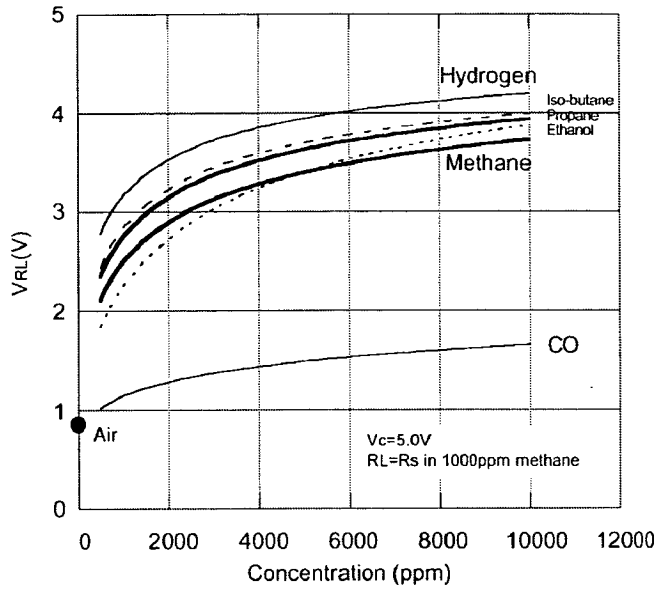


Figure 1.5 TGS813 responses to various analyte gases (note: responses are inverse of actual (resistance) response due to voltage divider circuit used to convert resistance into voltage; from Figaro [8])

The responses plotted in Figure 1.5 are the output of a voltage divider circuit where the voltage across a fixed resistor in series with the TGS813 sensor is measured using a voltmeter (this technique is recommended by Figaro to minimise current flowing through the sensing element and causing heating or damage). Inconveniently, the response is inverted since the voltage is measured across the series resistor, not the sensor itself (note: lower resistance = larger analyte response). A detailed discussion of the problems associated with this response measurement technique is presented in Chapter 2.

The ambient air resistance of SnO_2-based commercial sensors (as used in this work) when operated at recommended heater voltages is typically in the range of 15 to 30 kΩ. As explained in Section 1.2.3, this resistance value will fall non-linearly with increasing analyte gas concentration (values as low as ~100 Ω can occur for very high concentrations). To plot this multiple order-of-magnitude change in resistance, log-log plots are often used ($R = kp^{-n}$ is equivalent to $\log R = \log k - n \log p$). Such plots are of little analytical value but do provide a convenient means of visually comparing the sensor response to different analytes. Figure 1.6 shows the log-log plots published by Figaro [8] in their technical data sheet for the TGS813 sensor.

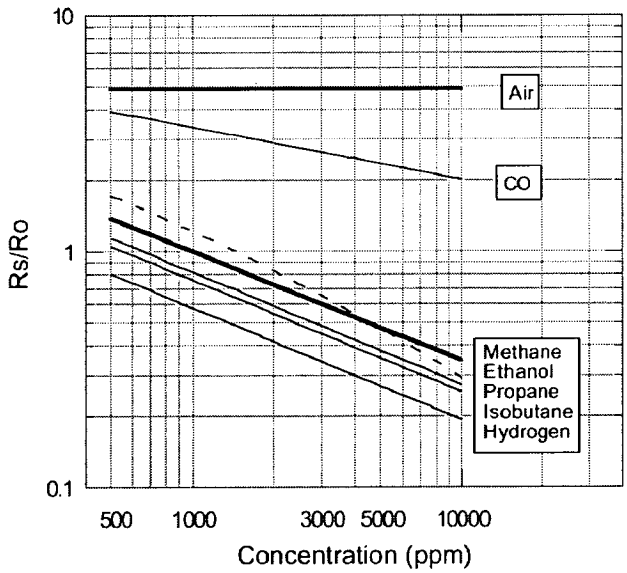


Figure 1.6 Log-log plots of TGS813 sensitivity to various analyte gases (R_S = sensor resistance, R_0 = sensor resistance in 1000 ppm methane; from Figaro [8])

The responses shown in Figure 1.6 are ratioed to the response of the sensor in a 1000 ppm methane/air mixture. One of the problems with metal oxide gas sensor design and manufacture is the difficulty in constructing two sensors of the same type, with identical response characteristics. Although the thick film techniques used in modern sensor design have helped to improve the 'consistency' of sensor manufacture, the problem remains. Fortunately, it can be minimised by using the ratio of the response of a particular sensor to its response at a known concentration of a given analyte. As can be seen in Figure 1.6, Figaro use this technique when they publish technical data for their sensors. This ratio of responses is commonly referred to as the 'sensitivity' and has been widely used by researchers when comparing responses of sensors constructed using different preparations of sensor material [14].

The interesting thing to note from both Figure 1.5 and 1.6 is the number of different analytes a single sensor will respond to. This lack of selectivity is probably the greatest drawback of using metal oxide sensors for gas detection and so has attracted the most research and development effort in recent years. Much of this 'selectivity' research has focused on sensor temperature modulation, or on the preparation of the sensing material, including the use of 'dopants' or promoters or combinations of both.

1.2.5 Temperature dependence of response

As mentioned previously, the key process in the response of an SnO_2-based sensor involves modulation of the concentration of surface-adsorbed oxygen species. By withdrawing electron density from the SnO_2 surface, adsorbed oxygen increases the sensor surface resistance. Reducing gases decrease surface oxygen concentration and thus decrease sensor resistance.

The temperature dependence of this process arises from three phenomena: (i) the differing stabilities of the surface oxygen species over the surface temperature range, (ii) the variation in rates of adsorption/desorption of oxygen and analyte with temperature, and (iii) the variation with temperature of the rate of reaction between surface-adsorbed oxygen and analyte gas [15]. While the identity of surface oxygen species remains slightly controversial [16-19] (discussed in greater detail in Chapter 4), this combination of effects results in the existence of optimum oxidation temperatures for different reducing gases. These optimum oxidation temperatures give rise to characteristic resistance-temperature (*R/T*) profiles when sensor resistance is plotted against temperature for a fixed analyte concentration. Figure 1.7 shows some typical sensitivity (resistance) - temperature plots for various analytes, highlighting the characteristic profiles.

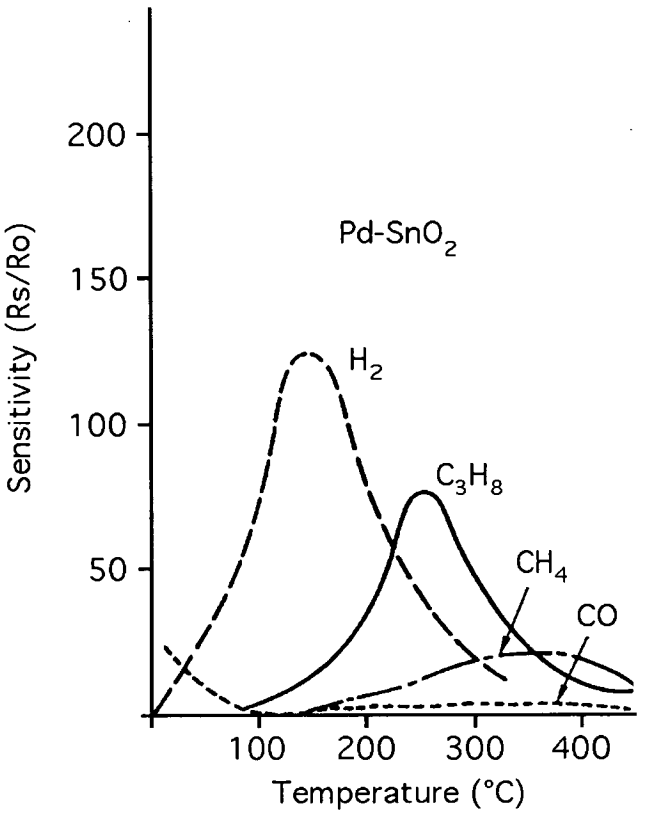


Figure 1.7 Sensitivity-temperature profiles for a 'typical' SnO_2 sensor in the presence of various analyte gases. Gas concentrations in air are 0.8 % H_2, 0.5 % CH_4, 0.2% C_3H_8, 0.02 % CO, (from Yamazoe and Miura [14])

Manufacturers such as Figaro use the optimum oxidation temperature effect to design sensors with some degree of selectivity. With analyte gases such as CO or alcohols which exhibit optimum oxidation temperatures lower than those of alkanes, selective sensors can be manufactured by carefully designing heating elements so that the sensor temperature matches that of the target gas optimum oxidation temperature. For example, Figaro sensors designed for alcohol detection typically have heater power consumptions which are two-thirds that of alkane-specific sensors [20,21].

1.2.6 Dopants and promoters

It was the original work by Shaver and Loh [4,5] which revealed the promoting effects of noble metals in gas sensor design. Inclusion of noble metals into the sensor material is not in fact 'doping' in the sense of generating *p*- and *n*- type semiconductors (amount of dopant in sensors is typically > 0.5 % compared with p- and n- type semiconductor doping of < 1 ppm [22]). However, the metal 'dopants' do alter the sensing characteristics of the base material. Although the addition of dopants to sensor material influences its response in a complex manner [14], three trends are generally observed:

- sensitivity is usually increased
- optimum oxidation temperature for the analyte gas in question decreases
- rate of response increases

It is the increase in sensitivity and corresponding decrease in optimum oxidation temperature which improves sensor selectivity to a particular gas. With noble metals widely used as catalysts for the oxidation of hydrocarbons, CO and H_2, it is not surprising that most commercial sensors targeting these gases include trace (< 1%w/w) amounts of either Pt, Pd or Ag, and in the case of SnO_2-based sensors, Pd in particular [14]. Figure 1.8 shows the effect on sensitivity and optimum oxidation temperature of including various noble metals in SnO_2-based sensors.

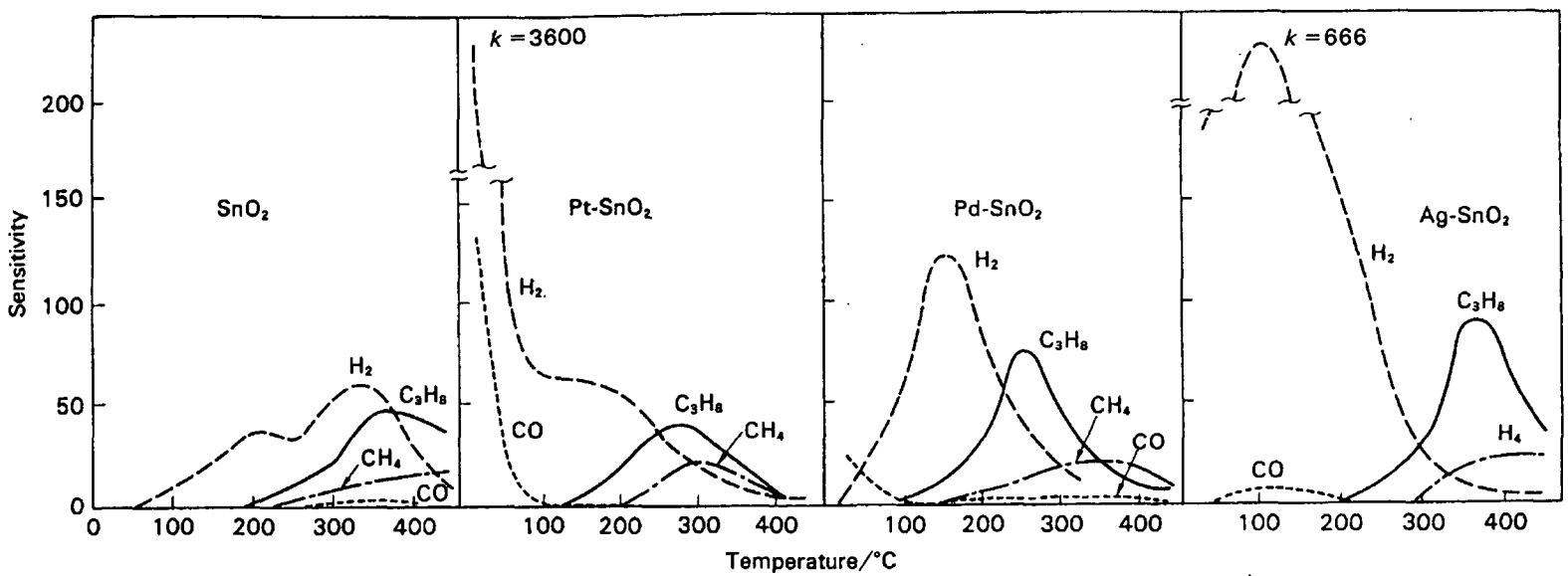


Figure 1.8 Sensitivity-temperature profiles for 'doped' and 'undoped' SnO_2 sensors in the presence of various analyte gases. Gas concentrations in air are 0.8 % H_2, 0.5 % CH_4, 0.2% C_3H_8, 0.02 % CO (from Yamazoe and Miura [14]).

From Figure 1.8 it is obvious that the SnO_2-based sensor with 0.5 wt % Pd loading provides the best separation of optimum oxidation temperatures and thus the best chance of improving selectivity through control of sensor temperature. Not surprisingly, the Figaro TGS813 sensor (designed for natural gas detection) includes ~ 1 % w/w Pd and operates in the region of 400 °C [23]. The other thing that is apparent from examining Figure 1.8 is the overlap of *R/T* profiles for various analytes, no matter what metal 'dopant' is used. It is this overlap that highlights the limitation of using 'dopants' and temperature control to improve selectivity.

Without a better understanding of the underlying response mechanisms, applications of metal oxide sensors will remain limited.

1.3 Where to from here?

An overview of metal oxide sensor technology reveals that over the last thirty years, a large amount of research and development effort has produced a range of versatile, low cost sensors for combustible gas detection [24]. However, as Yamazoe himself said in his definitive review paper, 'the fundamental understanding of the semiconductor gas sensor remains far from being satisfactory' [14]. Although that comment dates from the early 1990's, it has remained relevant until this work.

The following chapters describe research into the fundamental mechanisms responsible for the diverse response behaviour of metal oxide gas sensors, particularly SnO_2-based devices.

References

[1] P.T. Moseley, New trends and future prospects of thick- and thin-film gas sensors, *Sens. Actuators B* **3** (1991), 167-174.

[2] T. Seiyama, A. Kato, K. Fujiishi, M. Nagatani, A new detector for gaseous components using semiconductive thin films, *Anal. Chem.* **34** (1962), 1502-1503.

[3] N. Taguchi, Japan Pat. 45-38200 (1962).

[4] P.J. Shaver, Activated tungsten oxide gas detectors, *Appl. Phys. Lett.* **11** (1967), 255-257.

[5] J.C. Loh, Japan Pat. 43-28560 (1967).

[6] Figaro Engineering Inc., Figaro gas sensors 100, 500 and 800 series product catalogue, (1995).

[7] N. Taguchi, Gas Detecting Device, US Pat. 3695848 (1972).

[8] Figaro Engineering Inc., Technical information for TGS813, http://www.figarosensor.com/ (1999).

[9] Capteur Sensors and Analysers Ltd., Gas sensor technology, http://www.capteur.co.uk/ (2000).

[10] Figaro Engineering Inc., Figaro gas sensors 2000 series product catalogue, (1996).

[11] V.E. Henrich, P.A. Cox, in *The Surface Science of Metal Oxides* (1994), Cambridge University Press, Cambridge, pp. 4-7.

[12] W. Göpel, K.D. Schierbaum, SnO_2 sensors: current status and future prospects, *Sens. Actuators B* **26-27** (1995), 1-12.

[13] V.E. Henrich, P.A. Cox, in *The Surface Science of Metal Oxides* (1994), Cambridge University Press, Cambridge, pp. 153-155.

[14] N. Yamazoe, N. Miura, Some basic aspects of semiconductor gas sensors, in S. Yamauchi (ed.), *Chemical Sensor Technology*, Vol. 4 (1992), Kodansha, Tokyo, pp. 19-42.

[15] A.P. Lee, B.J. Reedy, Temperature modulation in semiconductor gas sensing, *Sens. Actuators B* **60** (1999), 35-42.

[16] S.R. Morrison, Selectivity in semiconductor gas sensors, *Sens. Actuators* **12** (1987), 425-440.

[17] P.T. Moseley, D.E. Williams, Oxygen surface species on semiconducting oxides, in: P.T. Moseley, J.O.W. Norris and D.E. Williams (eds.), *Techniques and Mechanisms in Gas Sensing* (1991), Adam Hilger, Bristol, pp. 46-60.

[18] W.M. Sears, K. Colbow, F. Consadori, General characteristics of thermally cycled tin oxide gas sensors, *Semicond. Sci. Tech.* **4** (1989), 351-359.

[19] W.M. Sears, K. Colbow, F. Consadori, Algorithms to improve the selectivity of thermally-cycled tin oxide gas sensors, *Sens. Actuators* **19** (1989), 333-349.

[20] Figaro Engineering Inc., Technical information for TGS822, http://www.figarosensor.com/ (1996).

[21] Figaro Engineering Inc., Technical information for TGS2620, http://www.figarosensor.com/ (1999).

[22] D.C. Giancoli, in *Physics for Scientists and Engineers* (1989), Prentice-Hall, New Jersey, pp. 959-960.

[23] Figaro Engineering Inc., The basis of Figaro gas sensor (1998).

[24] M.A. Craven, J.W. Gardner, P.N. Bartlett, Electronic noses-development and future prospects, *Trends Anal. Chem.* 15 (1996), 486-493.

Chapter 2

Equipment

2.1 Introduction

This chapter provides detailed information on the various pieces of experimental equipment used in the following chapters. The sensors used for temperature measurement, surface and response studies are described with the aid of schematics and ESEM (Environmental Scanning Electron Microscope) images, along with the results of Electron Probe Micro Analysis (EPMA) of the sensor surfaces. A schematic and description of the Fourier Transform Infrared Spectrometer (FTIR) instrument used for the new sensor temperature measurement technique (Chapter 3) is presented, along with details of the custom-made gas cell used for IR studies of the atmosphere above a sensor (Chapter 4). The design of a gas mixing and distribution manifold constructed for the preparation of gas mixtures is presented with a brief discussion of its features and operation. Finally, the electronics and software of a sophisticated data acquisition and control system built for sensor response measurements are described in detail.

2.2 Gas sensors

Indirectly heated (TGS813) and thick film (TGS2611) metal oxide gas sensors for general combustible gas detection, manufactured by Figaro, were used in all of the studies. Figaro sensors were chosen due to their reputation of operational reliability, repeatability, robust construction and their extensive use in previous research.

2.2.1 TGS813

The TGS813 sensor is an indirectly heated sensor of the original Taguchi design. It was designed for the detection of general combustible gases from 500-10,000 ppm in air [1]. Tin oxide 'doped' with < 1 % (w/w) palladium is sintered onto the outside of an alumina tube with a gold electrode positioned at either end [2]. A thin porous layer of SiO_2 provides mechanical protection of the sensor surface. A tightly coiled heating wire (60 μm) passing through the alumina tube provides indirect heating of the sensor surface. Heater and sensor electrical connections are made via six pins mounted in the sensor base. The base and cap are made of Nylon 66 with the openings in each covered with a double layer of 100 mesh 316 stainless steel gauze to prevent ignition of explosive gas mixtures. Figure 2.1 shows the construction of the TGS813 sensor and sensing element.

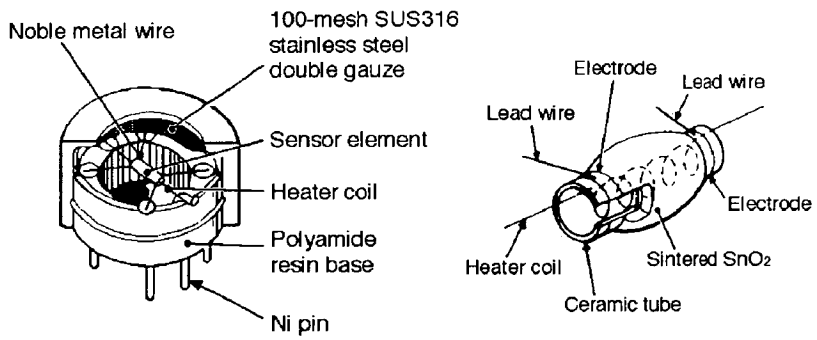


Figure 2.1 TGS813 sensor construction (from Figaro [2])

Figure 2.2 is a 20X ESEM (Environmental Scanning Electron Microscope) image of the sensing element clearly showing its structure and the gauze-covered opening in the sensor base. The lead wire arrangement provides duplicate electrical contacts to the sensing material and mechanical support when the sensing element is mounted in its base.

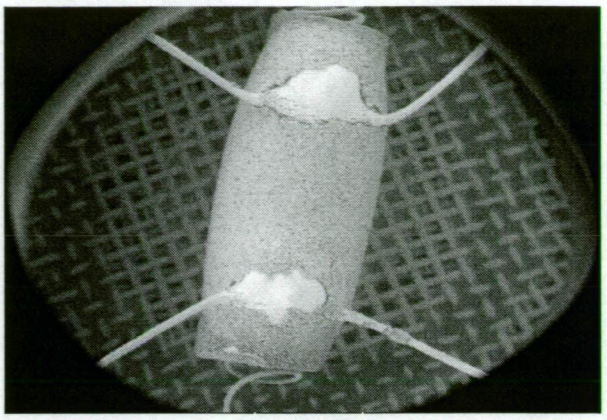


Figure 2.2 TGS813 Sensing element (20X)

A 400X ESEM image of the sensing element is presented in Figure 2.3. The darker coloured angular objects are pieces of alumina with the remainder of the surface consisting of SnO_2, SiO_2 and palladium (confirmed using EPMA, see below). According to Figaro, the cracks in the surface (porous layer of SiO_2), 'do not affect sensor performance' [3].

Figure 2.3 TGS813 Sensing element (400X)

An EPMA spectrum of the TGS813 surface is presented in Figure 2.4. The various X-ray emission lines have been labeled with their corresponding elements.

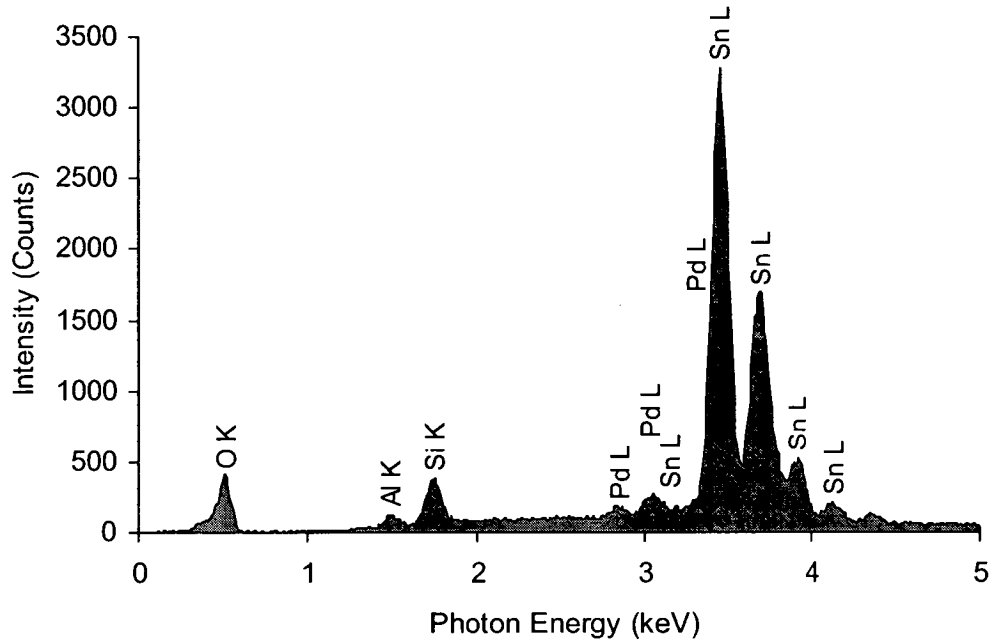


Figure 2.4 TGS813 surface EPMA (Electron Probe Micro Analysis) spectrum

Element	*Weight %*	*Atom %*
O K	29.8	72.0
Al K	1.0	1.5
Si K	3.8	5.2
Pd L	2.2	0.8
Sn L	63.2	20.5
Total	100.00	100.00

Table 2.1 Semi-quantitative elemental analysis of TGS813 surface (EPMA)

Percent normalised integration results for the spectral lines in Figure 2.4 are presented in Table 2.1. For the area analysed, the dominant material is SnO_2 with

the remainder consisting of alumina, SiO_2 and traces of palladium. Although this technique used no standards and therefore is semi-quantitative, the results verify the material published by Figaro [2,3] on sensor composition; i.e. that the TGS813 is primarily made of SnO_2 with a quantity of SiO_2 used to provide mechanical protection of the sensor surface and attenuate sensor response. Of particular interest is the trace amount of palladium, which is often used as a 'dopant' in construction of metal oxide gas sensors [4].

Several TGS813 sensors were used in the development of the sensor temperature measurement technique (Chapter 3). For the detailed sensor response work (Chapters 4 and 5), the more modern thick film TGS2611 device was used.

2.2.2 TGS2611

Like the TGS813, the TGS2611 sensor was also designed for the detection of general combustible gases from 500-10000 ppm in air. However, by using thick film printing techniques, the TGS2611 sensor offers greater uniformity between devices, miniaturisation and lower power consumption than the TGS813 [5]. The sensor element consists of a thin wafer of alumina substrate 1.5 × 1.5 mm onto which four gold contacts are printed. These provide electrical connection to the sensing material and heater. Two of the contacts extend through the substrate where they connect to two gold electrodes over which the 'doped' tin oxide sensor material is printed. A heater is printed onto the reverse (contact) side of the wafer and connected to the remaining two gold contacts. Platinum/tungsten alloy wires extend from the gold contacts to nickel/iron pins, which pass through the base of the sensor. The sensor base is made of nickel-plated steel while the cap is made of nickel/copper-plated steel. Gases diffuse into the sensor through an opening in

the cover. A double layer of 100 mesh 316 stainless steel gauze in the opening prevents ignition of explosive gas mixtures. Figure 2.5 shows the structure of the TGS2611 sensor and sensing element (note that the sensing surface faces the base of the sensor).

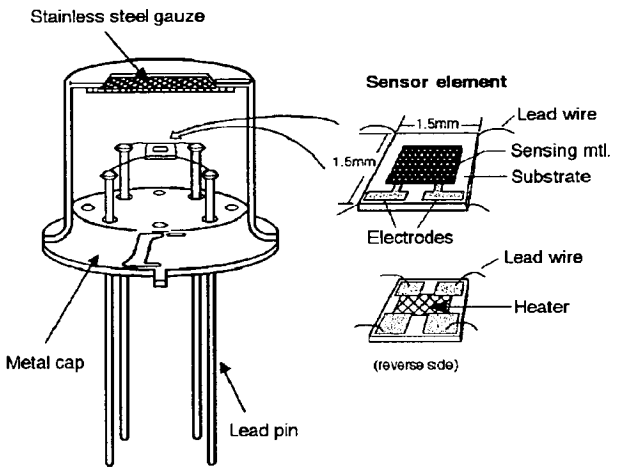


Figure 2.5 TGS2611 sensor construction (from Figaro [6])

Figure 2.6 is a 40X ESEM image of the TGS2611 sensing element, showing the gold electrical contacts and connecting wires. The heating element (not visible in this image) is positioned between the four contacts. Its electrical connection is through the two contacts at the bottom of the image.

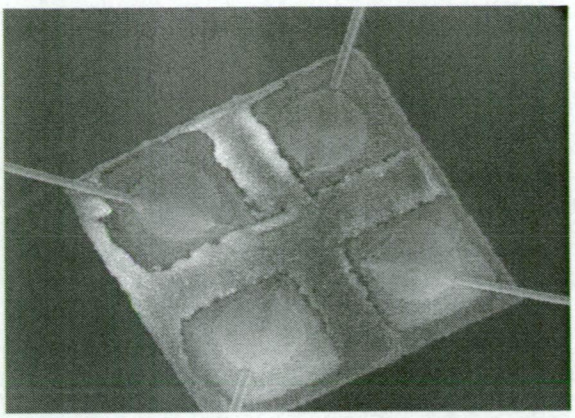


Figure 2.6 TGS2611 sensing element (contact/heater side) 40X

A 36X image of the TGS2611 sensing surface is presented in Figure 2.7. The sensor was removed from its base to clearly show the sensing material and contact arrangement (note the plated through-holes connecting the electrodes to the contacts on the reverse side).

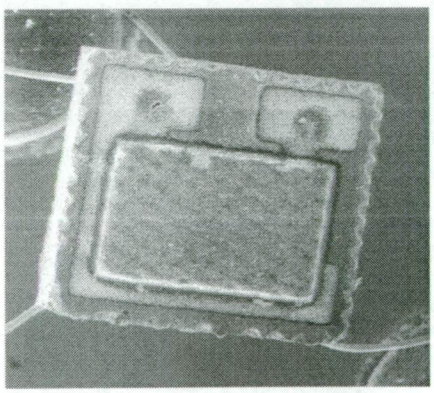


Figure 2.7 TGS2611 sensing element (sensor side) 36X

Figure 2.8 is an 1153X image of the TGS2611 sensor surface. The sintering of the uniform sensor material particles is clearly seen (note the presence of cracks in the surface as for the TGS813).

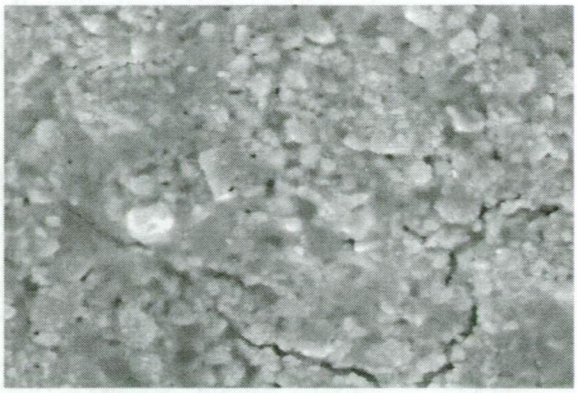


Figure 2.8 TGS2611 sensing element (1153X)

Figaro, the manufacturer of the TGS2611 sensor, provides little technical detail on the composition of the sensing material, other than saying that it is SnO_2-based. To confirm this and to identify any 'dopants', an EPMA of the sensor surface was conducted as for the TGS813. Figure 2.9 shows the EPMA spectrum of the TGS2611 surface. As before, the various spectral lines have been labelled with their corresponding elements.

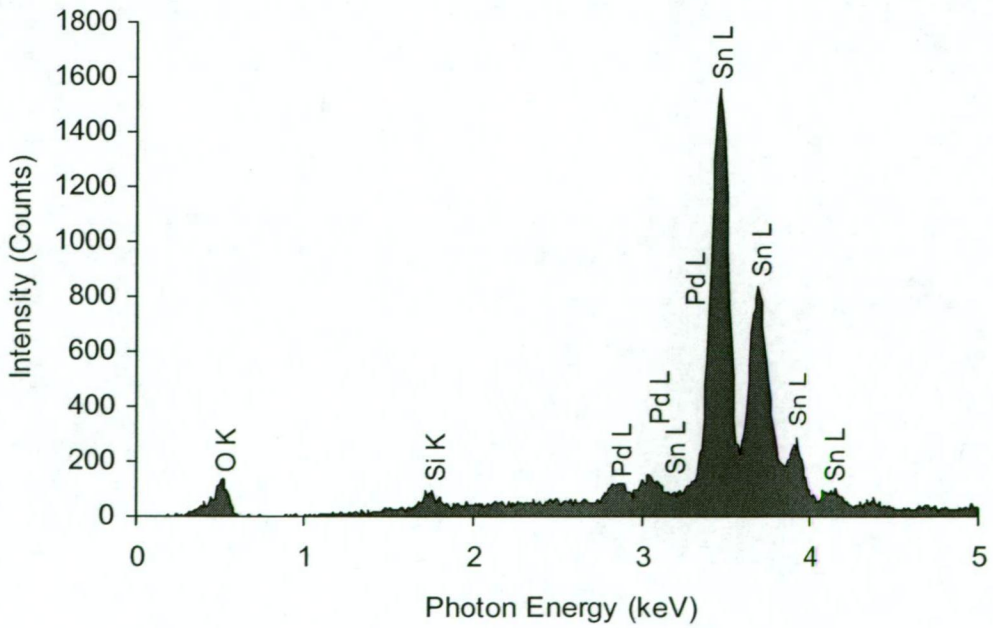


Figure 2.9 TGS2611 surface EPMA (Electron Probe Micro Analysis) spectrum

It is interesting to note, by comparing the spectrum in Figure 2.9 with Figure 2.4 for the TGS813, the similarities between the sensing materials. However, the TGS2611 spectrum does not indicate the presence of alumina, suggesting that the thick film printing process results in better separation of the sensing material from the substrate.

Element	*Weight %*	*Atom %*
O K	22.0	66.5
Si K	1.4	2.4
Pd L	0.00	0.00
Sn L	76.6	31.2
Total	100.00	100.00

Table 2.2 Semi-quantitative elemental analysis of TGS2611 surface (EPMA)

Table 2.2 contains the percent normalised integration results for the spectral lines in Figure 2.9. As with the TGS813 analysis, no standards were used and so the results are semi-quantitative. The analysis indicates an atomic ratio of tin to oxygen, which is much closer to that in pure SnO_2 than that found for the TGS813. Presumably, the thick film printing technique using for laying down the sensing material of the TGS2611, and the overall construction of the device, enable the use of less SiO_2 than in the traditional Taguchi style element of the TGS813. Although the table indicates no palladium present in the sample, comparison of this spectrum with that of the TGS813 indicates at least a fraction of one percent Pd present (see Pd $L\alpha_1$ & $L\alpha_2$ peak at ~2.83 keV). Once again it is interesting to note that palladium is used as the 'dopant' in this sensor.

As occurred with the TGS813, several TGS2611 sensors were used in the work presented in the next chapter (development of the sensor temperature measurement technique). More importantly, due to the excellent and reproducible response characteristics of this device, many TGS2611 sensors were used to provide real response data for testing and verification of the proposed models and simulations presented in Chapters 4 and 5.

2.3 Fourier Transform Infrared Spectrometer (FTIR)

2.3.1 Spectrometer

A Biorad FTS40 FTIR spectrometer was used as the IR detector for the radiometric sensor temperature measurement technique described in the Chapter 4 and for the analysis of the atmosphere above a TGS2611 sensor operating at various temperatures (Chapter 5). The spectrometer was fitted with an ambient temperature DTGS detector, a KBr beamsplitter and a ceramic IR source. This configuration ensured high IR throughput for both sensor IR emission (used to determine sensor temperature) and gas IR absorbance studies without the cost or complexity of a cooled detector and associated optics. Figure 2.10 is a simplified schematic of the FTS40 IR bench layout, showing the laser, IR beam path and gas cell (Section 2.3.2) which was used for the work in Chapter 5.

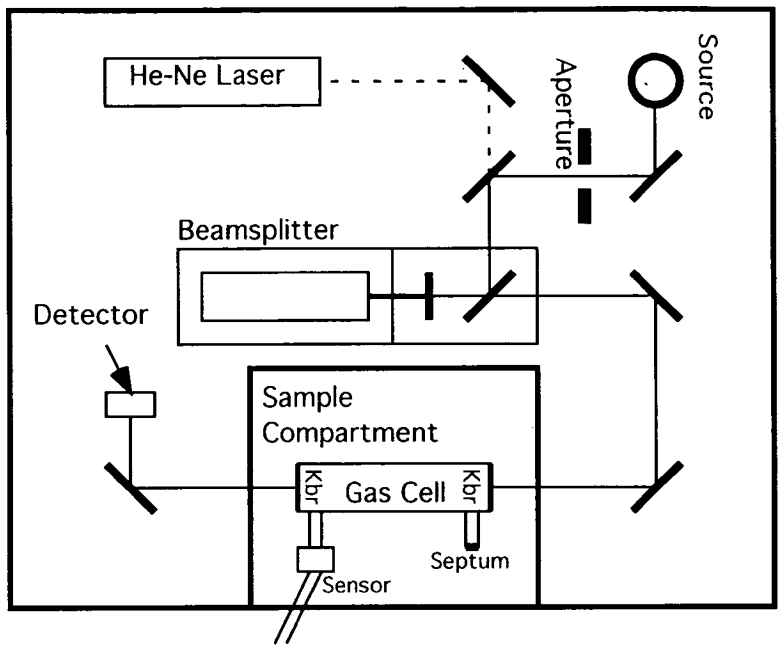


Figure 2.10 FTS60 IR bench schematic

2.3.2 IR gas cell

To facilitate the analysis of the atmosphere above a TGS2611 sensor operating at various temperatures (Chapter 5), a custom gas cell was made. This cell consisted of a glass tube fitted with KBr windows at both ends and two 10 mm diameter ports perpendicular to the main barrel (Figure 2.10). The sensor was connected to one port via a short piece of flexible tubing (for easy removal so that cell could be flushed), while a septum was fitted in the second port through which analyte samples were injected.

2.4 Gas mixing/dilution system

2.4.1 Design

A custom gas manifold system was designed and built to enable the mixing and dilution of gases for sensor response studies (presented in Chapters 4 and 5). The overall layout of this system was based on traditional Schlenk-type manifolds. Extra connections and joints were included in the design to enable easy modification if additional functions were required as the research progressed. Figure 2.12 is the engineering schematic of the manifold system.

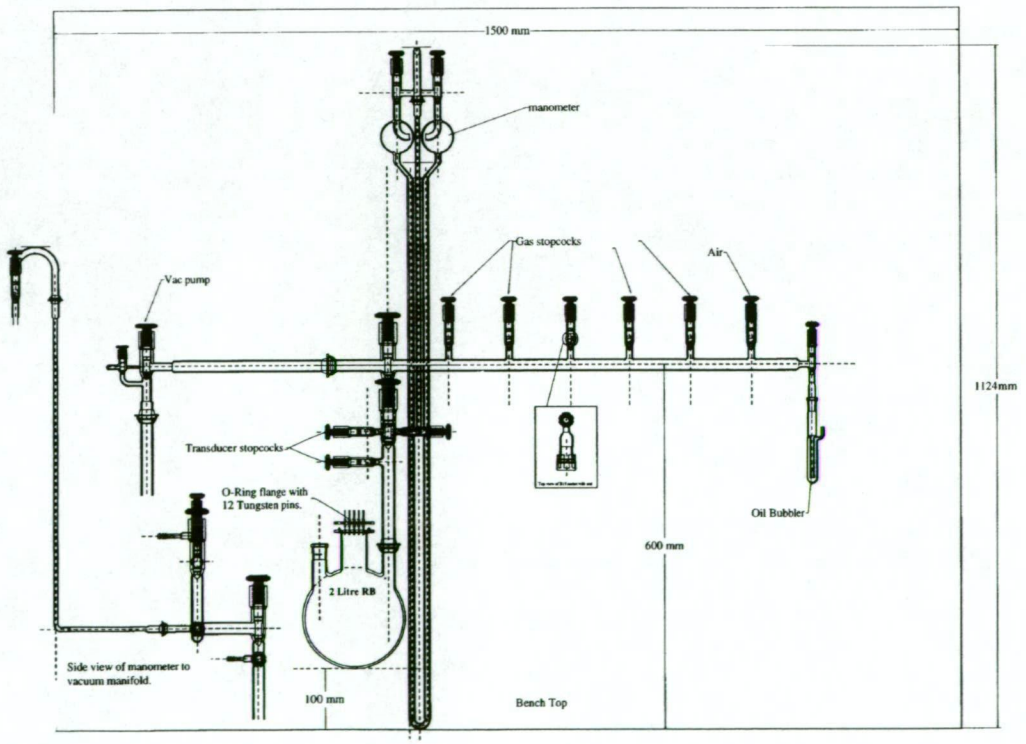


Figure 2.12 Gas mixing/dilution manifold schematic (from Mike Brandon [7])

To minimise the entry of impurities into the gas mixtures, construction materials were carefully chosen. Glass was used for the construction of the main tubes, joints, taps and sample chamber to minimise any out-gassing or gas adsorption problems.

The main manifold line was constructed in two pieces, joined with o-ring type ball joints to allow for any misalignment. A large bore tap and bleed valve was used to form the first piece, connecting the vacuum pump (Edwards RV3, max. vac: 1×10^{-3} mmHg [8]) to the main line which consisted of a large bore (20 mm diameter) tube. At the vacuum pump end of this tube was the first isolation tap connecting the sample chamber branch line to the main line, at successive intervals after this were hose barb terminated taps for connection of Tygon 2275 6 mm ID tubes from the gas cylinders and air supply. One of the main line gas taps was fitted with a female B19 ground glass conical joint to either hold a septum or other fittings if necessary. Furthest away from the vacuum pump was an oil-bubbler for venting gas and thus maintaining manifold gas pressure at 1 atm.

The main sample compartment consisted of a 2.34 L flask with three necks. The right side neck terminated in an o-ring-type ball joint to facilitate connection to the manifold via two isolation taps. The central neck terminated in a flange with an opening of ~40 mm, through which the circuit board holding the sensors was lowered into the sample chamber. Finally, the left side neck terminated in a female B19 ground glass conical joint, which was used to hold a septum (mounted in a fitting for easy removal) through which small volumes of gas could be admitted via syringe. The entire flask was submerged in a water bath of temperature 25 ± 0.5 °C.

Two pressure measuring devices were attached to the branch tube connecting the sample flask to the main manifold tube. A mercury-filled manometer was used for direct measurement of flask pressure, while a solid state absolute pressure transducer (SensorTechnics, HCX001A6V [9]) provided a 0.5 – 4.5 V signal (proportional to flask pressure) to the data acquisition system described in Section 2.5. The solid state transducer was connected to the transducer stopcock immediately below the first isolation valve while the manometer was connected to one of the transducer stopcocks mounted between the isolation valves (see Figure 2.12). This arrangement was used to enable the data acquisition system to accurately measure flask pressure without compensating for the dynamic volume problems associated with the manometer.

A glass disc was made to facilitate electrical connection between the sensor under test and the data acquisition system described later. Twelve tungsten pins were embedded in the glass, onto which copper connecting wires were silver-soldered. This system ensured there were no out-gassing problems associated with making gas tight electrical connections using adhesives.

Gas sensors were mounted on a custom made circuit board via sockets (Figure 2.13). The circuit board itself was made from high grade FR4 epoxy glass to minimise out-gassing problems. A 25 mm diameter brushless, sealed bearing fan was mounted at the bottom of the circuit board to provide mixing of the sample chamber contents.

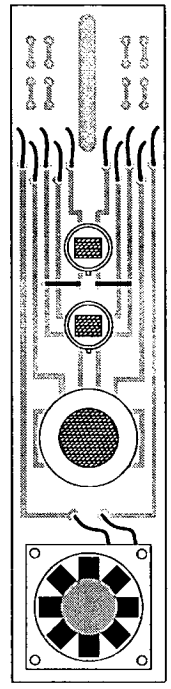


Figure 2.13 Sensor circuit board (note: hangs vertically in sample chamber)

2.4.2 Gases and analytes

To provide a source of clean, fixed-humidity air, compressed air from the lab supply was firstly filtered, then passed through a column of activated charcoal, bubbled through a sodium chloride solution and finally a filter trap before being admitted to the manifold. This arrangement was chosen after much experimentation, and supplied air with a water vapour partial pressure of ~16 mmHg. Oxygen response studies (see Chapter 4) were carried out using high purity grade gas supplied in 'G' sized cylinders (BOC 7600 litres). Alkanes (methane – butane) used for the response work (see Chapter 5) were of GC grade, supplied in bottles (Alltech 14 litres) fitted with regulators and syringe adaptors for dispensing. The remaining liquid (at 25 °C) alkanes were of AR grade (Sigma-Aldrich).

2.4.3 Operation

Two main techniques were used to prepare gas samples for sensor response studies. For single gas (e.g. oxygen) response studies, the sample chamber was firstly evacuated via the main manifold line. The sample chamber was then sealed using the lower isolation tap (this enabled the solid state sensor to monitor sample chamber pressure, while the manometer monitored the manifold line pressure). With the vacuum pump isolated, gas was then admitted into the main line and vented out through the oil-bubbler. To then slowly fill the sample chamber, gas was transferred from the main line to the sample flask via a Tygon tube fitted with needles at either end which were inserted into septa in the B19 joints of the flask and the main line branch tap. By varying the diameter and length of the needles, the rate of pressure increase in the flask could be controlled.

Analyte sensor response studies involved the preparation of gas mixtures. The sample chamber was firstly flushed using clean air (see earlier) via the main manifold line, with this air exiting the flask through the B19 joint. With the flask then sealed, the mixture was formed by injecting analyte through the septum and fitting in the B19 joint. Both gaseous and liquid analytes were transferred in this manner, using appropriate syringes to minimise leakage and contamination. The analyte volumes required to generate the correct concentrations were calculated using a flask temperature of 25 °C and vapour pressures (for liquid analytes) obtained from the Antoine equation.

2.5 Data acquisition and control system - hardware

Many studies into the response mechanisms of tin oxide gas sensors have been hindered by the inability to accurately measure sensor resistance with respect to sensor surface temperature. With this limitation in mind, a sophisticated data acquisition and control system was designed and constructed with the following features :

- computer control of sensor temperature to within 5 K of target value
- ability to measure and store sensor resistance to within 1 % of the actual value over the resistance range 0 to 1 x 10^8 Ω
- real-time display of sensor resistance versus temperature
- expandability (so that other measurement devices, e.g. pressure transducers, could be easily connected)

To provide these features and minimise cost, it was decided that a microprocessor-based system would be built which could control sensor temperature while simultaneously measuring, processing and transferring input data to a personal computer for real-time display.

2.5.1 Microprocessor

To minimise development time, a proprietary micro-controller device was purchased from Onset Computer. This device, known as a Tattletale Model 8, is based around the Motorola 68332 32-bit processor running at variable clock speeds up to 16 MHz [10]. To simplify design of embedded systems using this device, the Tattletale has the following features:

- on board voltage regulation, 3.3 V and 5 V output (7-15 V input)
- 256 Kbytes of RAM
- 256 Kbytes of electrically erasable EPROM for program storage
- two RS-232C data comms ports for communication with a PC
- an eight channel 12-bit successive approximation analogue to digital converter (ADC) with a 0 to 4.096 V input range (LSB = 1 mV)
- twenty four user programmable, bi-directional digital input/output (I/O) lines.

The Tattletale has no on-board digital to analogue (D/A) converter for generating voltage waveforms, nor does it provide any form of signal-conditioning apart from basic low pass filtering. To generate the waveforms necessary for controlling sensor temperature, an interface printed circuit board (PCB) was constructed to provide the necessary D/A circuitry and buffering for driving the heating element of the sensor under test. A second signal-conditioning PCB was constructed to convert the resistance of the sensing element into a signal suitable for the Tattletale on-board ADC.

2.5.2 *Digital to analogue (D/A) interface board*

The D/A interface board was comprised of several circuits, including the power supply section, the serial to parallel shift register, the D/A converter, a voltage reference for D/A output and the D/A output buffer. Figure 2.14 shows the complete circuit schematic for the D/A interface board external to the Tattletale Model 8.

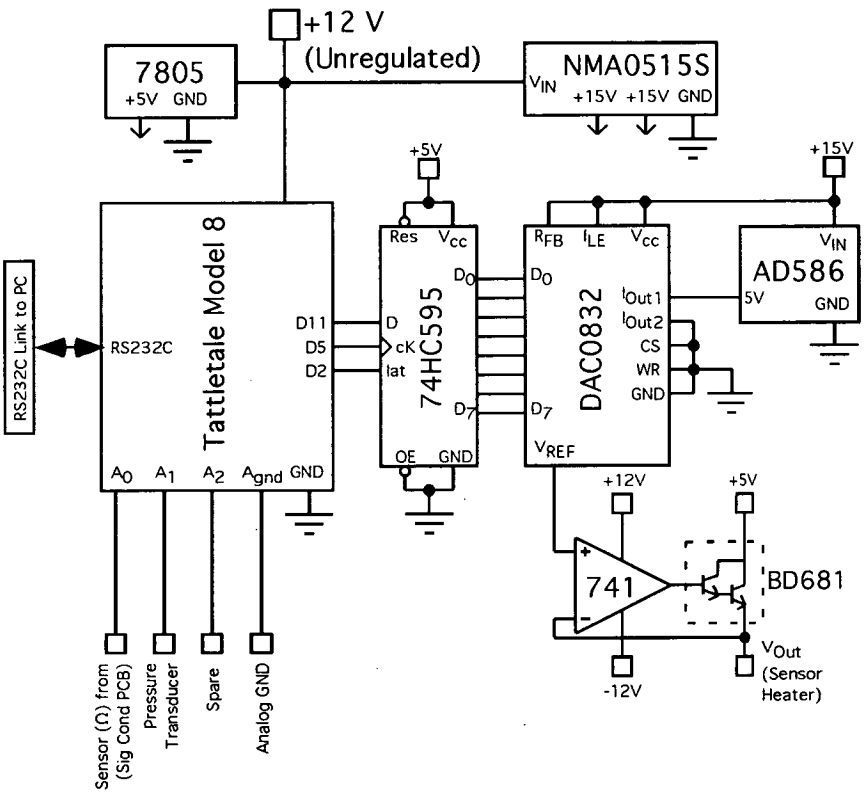


Figure 2.14 Interface board schematic

Power for the entire data acquisition system is supplied from an unregulated 12 V DC supply. The Tattletale's on-board regulators are directly connected to this supply, while the D/A interface board has its own LM7805 5 V regulator. It was decided to separate the power supplies in this manner (instead of deriving interface power from the Tattletale's regulators), so that in the event of a short

circuit occurring in a sensor heater element, no damage would be done to the Tattletale module. The D/A output buffer circuitry and D/A voltage reference required a dual rail +12 & -12 V supply. This was generated by a 1 W, 5 to ±12 V DC to DC converter (NMA0515S).

Three of the digital output lines of the Tattletale were configured to output serial data. Serial input D/A converters are not readily available, so a serial to parallel shift register was required. A 74HC595 [11] was used to provide the interface between the Tattletale serial output and the D/A converter parallel input. This device contains an 8-bit serial-in, parallel-out shift register connected to an 8-bit tri-state latch, the output of which was routed to the D/A integrated circuit (IC).

An 8-bit National Semiconductor DAC0832 D/A converter [12] was used to convert the digital data from the Tattletale into an analogue signal. The internal resistor ladder network of this device was connected in a voltage mode switching configuration (I_{Out} and V_{Ref} reversed, V_{Ref} = D/A output) so that the output voltage would be given by the following equation:

$$0V \leq V_{Out} \leq \frac{n}{256} V_{Ref}$$

where n is the digital value written to the D/A by the Tattletale via the 74HC595 shift register. Thus, the step size or least significant bit (*LSB*) of the D/A converter would be 0.0195 V for a voltage reference of 5 V. For a TGS2611 sensor operating from 298 to 766 K (0 to 5 V, Chapter 3), this step size yields a temperature increment of ~1.83 K per D/A step.

The output range and step size of the D/A converter was set by a 5 V precision reference voltage generator (AD586N [13]). This device uses a laser-trimmed ion-implanted Zener diode to generate a high precision 5.000 V ± 2 mV output over the temperature range 0 °C to 70 °C. Using such a stable voltage reference ensured that the sensor temperature fluctuations were minimised.

A typical gas sensor such as the TGS2611 has a heater resistance in the order of 50 Ω. At an applied heater voltage of 5 V, the current required is ~100 mA. With the D/A converter capable of sourcing a maximum of 10 mA (a limit imposed by the AD586 V_{Ref}), a buffering circuit was required to generate the necessary current for driving the heating element of the gas sensor under test. A JFET bipolar operational amplifier (1/2 LF352 op-amp) was connected in a voltage follower configuration with a BD681 40 W NPN Darlington pair transistor. The collector of the Darlington pair was connected to the 5 V rail, while the emitter was connected to the inverting input of the op-amp (to provide feedback and eliminate the base-emitter turn-on potential). Thus the voltage appearing at the emitter followed the op-amp non-inverting input from the D/A converter. The sensor heating element was connected from the Darlington pair emitter to ground.

2.5.3 *Signal conditioning board*

A major difficulty of tin oxide gas sensor research is accurately measuring the resistance of sensing elements. The resistance of a typical sensor can vary from $1 \times 10^8\ \Omega$ with the sensor at ambient temperature to hundreds of ohms in the presence of an analyte gas, with the sensor operating at a temperature for optimum response to that analyte. A typical approach to measuring sensor resistance and that recommended by Figaro (manufacturer of the sensors used in this work) is to use a voltage divider configuration as shown in Figure 2.15. This circuit produces a non-linear voltage output with respect to sensor resistance as shown by the equation:

$$V_{Out} = V_{In} \frac{R_S}{R_L + R_S} \tag{1}$$

The problem that occurs with such a system is that when the output from this circuit is input into an ADC, the resistance measurement error varies with the magnitude of the sensor resistance. For the detailed studies conducted in this research, an analogue circuit was designed to convert the logarithm (base 10) of the sensor resistance into a voltage signal (0 – 4 V) suitable for input into the Tattletale ADC circuit. With Tattletale software calculating the antilogarithm of the ADC input signal, the sensor resistance can be measured with a constant error over six orders of magnitude, thus overcoming the problems of the voltage divider system.

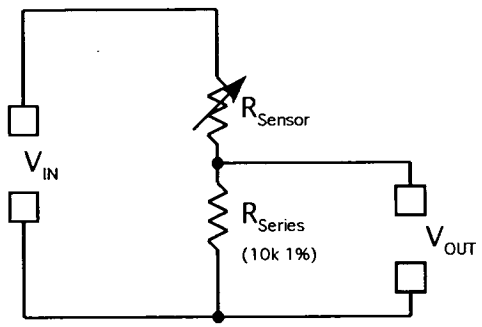


Figure 2.15 Voltage divider circuit for measuring sensor resistance

Practical logarithmic converter circuits make use of the logarithmic relationship between the emitter–base voltage of standard double-diffused transistors and their collector current from below 1 pA to above 1 mA [14]. Using a super matched pair of monolithic NPN transistors (LM194 [15]) and two high performance 741 op-amps the signal conditioning circuit board was constructed to convert a linear current to a logarithmic voltage.

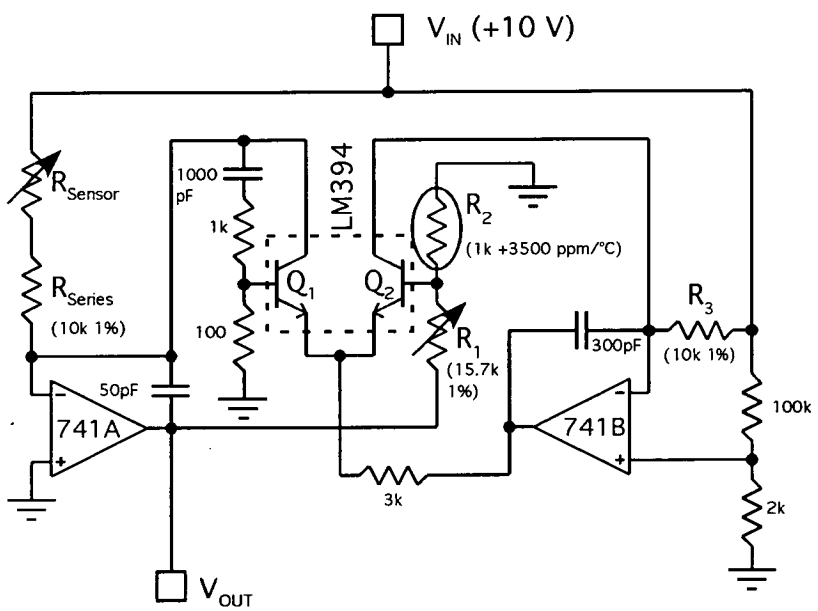


Figure 2.16 Signal conditioning board circuit schematic

The circuit for the signal conditioning board (Figure 2.16) was based on an original design by National Semiconductor [15]. Transistor Q_1 is used as the non-linear feedback (log) element around the LM741A op-amp. Negative feedback is applied to the Q_1 emitter through divider R_1 and R_2, and the emitter base junction of Q_2. This forces the collector current of Q_1 to exactly equal the current through the gas sensor and its fixed series resistor. Transistor Q_2 provides feedback for the second op-amp LM741B. Negative feedback forces the collector current of Q_2 to equal the current through R_3. Since the Q_2 collector current remains constant, the emitter-base voltage also remains constant and therefore the V_{BE} of Q_1 varies with a change in sensor resistance.

The output voltage of the circuit is a function of the difference in emitter-base voltages of Q_1 and Q_2:

$$V_{Out} = \frac{R_1 + R_2}{R_2}(V_{BE_2} - V_{BE_1}) \tag{2}$$

For matched transistors operating at different collector currents, the emitter-base differential is given by:

$$\Delta V_{BE} = \frac{kT}{q}\log_e \frac{I_{C_1}}{I_{C_2}} \tag{3}$$

where k is Boltzmann's constant, T is the temperature in kelvin and q is the charge on an electron.

Combining equations 2 and 3 produces the following equation for the output voltage:

$$V_{OUT} = \frac{-kT}{q}\left[\frac{R_1 + R_2}{R_2}\right]\log_e\left[\frac{V_{IN}R_3}{V_{IN}(R_{Sensor} + R_{Series})}\right], V_{IN} \geq 0 \text{ V} \quad (4)$$

The log term is directly proportional to absolute temperature and without compensation, the scale factor would also vary with temperature. However R_2 (1 kΩ, ±1 %, +3500 ppm/°C) is also directly proportional to temperature over the range –25°C to 100°C for the resistor shown and so the temperature dependence of the log term is cancelled and a constant scale is obtained.

For the component values given, the circuit produces the following relationship between sensor resistance and output voltage:

$$V_{OUT} = \log_{10}\left[\frac{10000}{R_{Sensor}(\Omega) + 10000}\right] \quad (5)$$

With the signal conditioning board performing the logarithmic conversion of sensor resistance to voltage, recovery of the actual sensor resistance data by software was a relatively straightforward programming task.

2.6 Data acquisition and control system – software

Two software packages were developed for the data acquisition and control system. The 'real-time package' consisted of two linked programs (one running on the Tattletale and the other on an Apple Macintosh (MAC)) providing real-time plotting and disk storage of sensor resistance data while the sensor temperature was varied. The second 'logging package' was a stand-alone program (running on the Tattletale and controlled via the Tattletale communication package on the MAC) which sampled sensor resistance and sample chamber pressure at user-selectable sampling rates while holding the sensor temperature static (user selectable).

The Tattletale was programmed using a sophisticated version of Basic (Tx-Basic). This language dispenses with line numbers and uses an assembler-style label method for implementation of loops, functions and procedures. Powerful instructions have been included to provide accurate timing, simple access to the ADC channels and to input and output digital data serially.

Onset Computer provides PC and MAC versions of the Tattletale development package. Basic programs are written using the in-built text editor, then compiled and downloaded to the Tattletale RAM via the RS232C link. An ASCII communications module in the package allows user interaction with active programs if desired. Program execution by the Tattletale occurs immediately after download, however the program remains in RAM only while power is supplied. Alternatively, the Tattletale can be instructed to burn the program into EEPROM, thus program execution occurs immediately upon power-up.

Metroworks Codewarrior was used to develop the real time display software running on the MAC. The program was written in Pascal and made extensive use of graphics for the real time plotting of resistance data.

2.6.1 Real time package

The real-time package Tattletale program was responsible for generating a triangle voltage waveform for the sensor heater while simultaneously sampling sensor resistance and sending this data as packets to the second program running on the MAC. The waveform minimum and maximum voltages and wavelength were user-selectable from the MAC program via a dialog box. Sampling rate and wavelength were linked with a fixed 256 samples per wave and 10 to 0.1 samples per second, producing wavelengths from 25.6 to 256 seconds. To improve the real-time display of resistance data, data packets contained the sample number and raw ADC data, and conversion into resistance values was performed by the MAC program. This scheme was chosen as it enabled real-time scaling of resistance plots. The MAC program provided a snapshot feature so that a real-time resistance plot could be compared with a historical plot. From one to ten resistance plots could be averaged and written to disk as a text file for data import into other packages for further analysis. Figure 2.17 is a screen dump of the main dialog window of the MAC program.

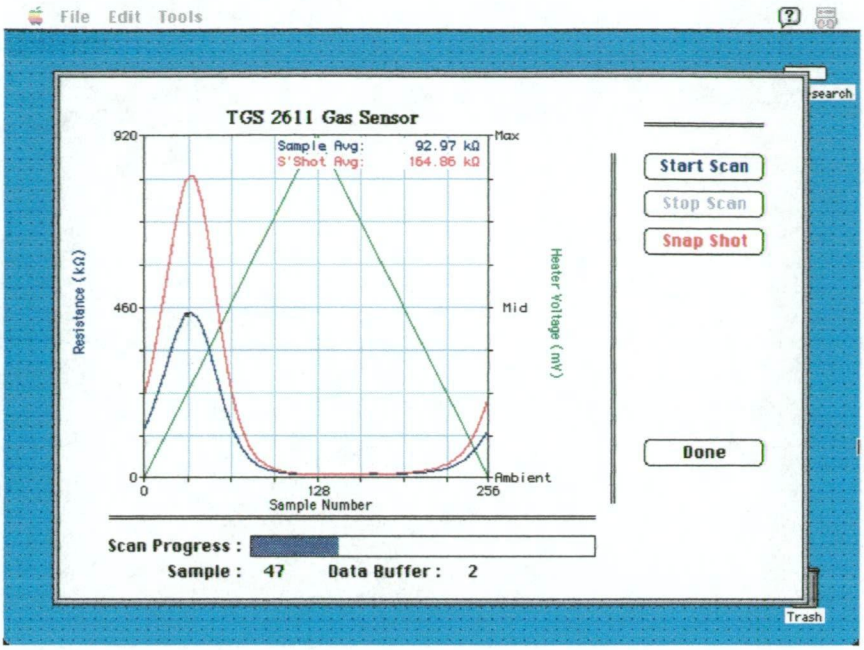


Figure 2.17 Real time package main window

2.6.2 Logging package

The logging package used the ASCII communications package for user interaction and display of data. Upon running the program, the user is asked to enter the sensor heater voltage (ie temperature) and the interval between samples. The program then waits until the user presses 'g', at which point the program repeatedly samples the sensor resistance and sample chamber pressure, sending the values back to the communication package for display on-screen as two columns of numerical data. This process continues until the user presses 's', at which point the user is again prompted for heater voltage and sampling interval, ready for the next run. The communications package had a data buffer capacity of 40 kbytes and so ~2000 samples could be stored before loss. Data transfer was accomplished by simply cutting and pasting the relevant text containing the data into a new text file, from which the data could be imported into other packages for analysis.

2.6.3 *Program listings*

Due to the complexity of the programs that comprise the real time and logging packages, a complete program analysis would be lengthy and beyond the scope of this thesis and so the complete code listings are included in Appendices 1-3 for any interested readers.

References

[1] Figaro Engineering Inc., Figaro gas sensors 100, 500 and 800 series product catalogue, (1995).

[2] Figaro Engineering Inc., Technical information for TGS813, http://www.figarosensor.com/ (1999).

[3] Figaro Engineering Inc., The basis of Figaro gas sensor, (1998).

[4] N. Yamazoe, N. Miura, Some basic aspects of semiconductor gas sensors, in S. Yamauchi (ed.), *Chemical Sensor Technology*, Vol. 4 (1992), Kodansha, Tokyo, pp. 19-42.

[5] Figaro Engineering Inc., Figaro gas sensors 2000 series product catalogue, (1996).

[6] Figaro Engineering Inc., Technical information for TGS2611, http://www.figarosensor.com/ (1999).

[7] M. Brandon, Gas mixing/dilution manifold schematic, (1998).

[8] Edwards High Vacuum International, RV3 Vacuum pump operators manual and specifications (1995).

[9] SensorTechnics, Data Sheet: HCX001A6V Fully signal conditioned pressure transducer (1998).

[10] Onset Computer Corporation, Tattletale Model 8 installation and operation manual (1995).

[11] National Semiconductor, Data Sheet: MM74HC595 8-bit shift registers with output latches (1988).

[12] National Semiconductor, Data Sheet: DAC0832 8-bit μP compatible, double buffered D to A converter (1995).

[13] Analog Devices, Data Sheet: AD586 High precision 5V reference (1993).

[14] National Semiconductor, Application Note 30: Log Converters (1969).

[15] National Semiconductor, Data Sheet: LM194 Supermatch pair (1994).

Chapter 3

Sensor temperature measurement

3.1 Introduction

As explained in Chapters 1 and 2, metal oxide gas sensors typically operate at elevated temperatures attained through application of a current to an internal heating element [1-3]. The non-linear response characteristics of these sensors are highly temperature dependent, with different reducing gases giving rise to characteristic resistance versus temperature profiles. Because of this, any substantive research into the underlying response mechanisms requires accurate measurement and control of sensor temperature [4]. Much of the work involving sensor temperature considerations published to date does not include actual measurements of the surface temperature of the sensor, does not specify how temperature measurements were made, or only infers or estimates sensor temperatures from other work. This is largely because of the difficulties involved with the measurement of the temperature of a small object (for example, the Figaro TGS26XX sensors are roughly 1.5 mm x 1.5 mm x 0.5 mm in size, see Chapter 2). It is for this reason that many workers choose to express the temperature of their sensors in terms of the applied heater voltage, even though the relationship between temperature and voltage cannot be assumed to be linear.

Some semiconductor gas sensors use platinum resistances embedded in the substrate material to measure the surface temperature, but the most commonly used commercial sensors do not avail themselves of this technique. Published methods for the temperature determination of these sensors fall into two main categories: (i) direct contact methods, which include the use of thermocouples or thermistors [5]; and (ii) the use of commercial infrared thermometers (IRTs) [6, 7]. Direct contact methods are not reliable for determining the temperature of

small objects because of problems with making good thermal contact, and because of heat transfer effects [8-10]. However, in unpublished work, Clifford [11] devised a two-thermocouple method in which the sensor temperature was determined by modelling the thermal gradient between the gas sensor surface and its "ambient" surroundings inside its case. The temperatures obtained by Clifford for the operation of a TGS812 sensor were significantly higher than most subsequent estimates, and prior to this work had not been independently verified.

The main problem with using infrared thermometers to measure gas sensor surface temperatures is that these devices must be able to focus on the sensor surface, without 'seeing' any surrounding objects. This means that owing to constraints imposed by the optics of the instrument, there will be a minimum size for the object being measured, otherwise the temperature will be underestimated. Another limitation of IRTs is that they assume a particular value for the total emissivity of the surface being measured, when the actual emissivity may be significantly different. The large discrepancy (up to 200 K) between sensor surface temperatures published by Figaro [1] and those published by various workers may be attributable to the limitations of the IRTs employed.

Some workers have attempted to estimate sensor temperature by measuring the conductance of the sensor (with no applied heater voltage) in an oven whose temperature can easily be measured and adjusted, and then relating this conductance to that of a sensor operating at given applied heater voltages [12]. This approach is flawed because the sensor is being operated in two very different atmospheric environments and thus can only give a very rough estimate at best of the sensor temperature. A related idea is to attempt to measure the change in

resistance of the heater wire (in indirectly heated Taguchi style TGS8XX sensors) with temperature (i.e. in an oven), and use the temperature coefficient of the wire metal to calculate absolute temperature. However, the temperature coefficient of the heater wire typically used in TGS8XX sensors is too low to give precisely measurable changes in resistance over the operational voltage range, and there is the implicit assumption that the sensor surface and the heater wire are in thermal equilibrium. Calorimetric techniques are not practicable owing to the difficulty in determining the precise thermal properties of the sensor. Raman Stokes/anti-Stokes intensity ratio techniques [13] for temperature determination are problematic because of the need to avoid heating of the sensor surface by the laser (and because of the instrumental requirements).

This chapter describes a practical method for the estimation of sensor surface temperatures using a Fourier Transform Infrared Spectrometer (FTIR), a very common laboratory instrument. The use of spectroscopic techniques to determine surface temperatures and/or emissivities is not new, and various methods have been published which permit surface temperature determination to within ± 1 K or less [10, 14-17]. However, the method of Clausen et al [10] necessitates the use of a cooled (sub-ambient) IR detector (with the added complications that this introduces), while the method of Markham et al [14] involves the fitting of data to functions which are very sensitive to instrumental noise. The aim of the work presented here was to develop a relatively simple and inexpensive method for determining the surface temperatures of commercial gas sensors to within about 10 K. Even with an uncertainty of this size, this information is of great practical and theoretical utility in metal oxide gas sensor research given the large discrepancies between published temperatures (as mentioned above) and the

difficulties created by the size and construction of commercial gas sensors. In addition to the method described, temperature estimates are presented for the two gas sensors (TGS813 and TGS2611, used in the research presented in subsequent chapters) operated at various applied heater voltages.

3.2 Theory and methodology

A blackbody is a theoretical entity that is a perfect absorber and emitter of radiation over all wavelengths. It can be simulated by maintaining an empty container (a cavity with insulating walls) at a constant temperature and monitoring the radiation that escapes through a pinhole in the container. The energy density, E (J m^{-3}), of blackbody radiation is given by the Stefan-Boltzmann law:

$$E = \int_0^\infty \rho d\lambda = aT^4 \qquad a = \frac{8\pi^5 k^4}{15c^3 h^3} \tag{1}$$

where k is the Boltzmann constant, h is the Planck constant and c is the speed of light in a vacuum. This is an integration of the Planck distribution, which gives the energy output as a function of wavelength, λ (and thus of frequency, $\nu = \frac{c}{\lambda}$):

$$\rho = \frac{8\pi hc}{\lambda^5}\left(\frac{1}{e^{hc/\lambda kt} - 1}\right) \tag{2}$$

This is the characteristic blackbody radiation curve, which has a maximum and a total energy density (E, the area under the curve) that vary with temperature, as seen in Figure 3.1.

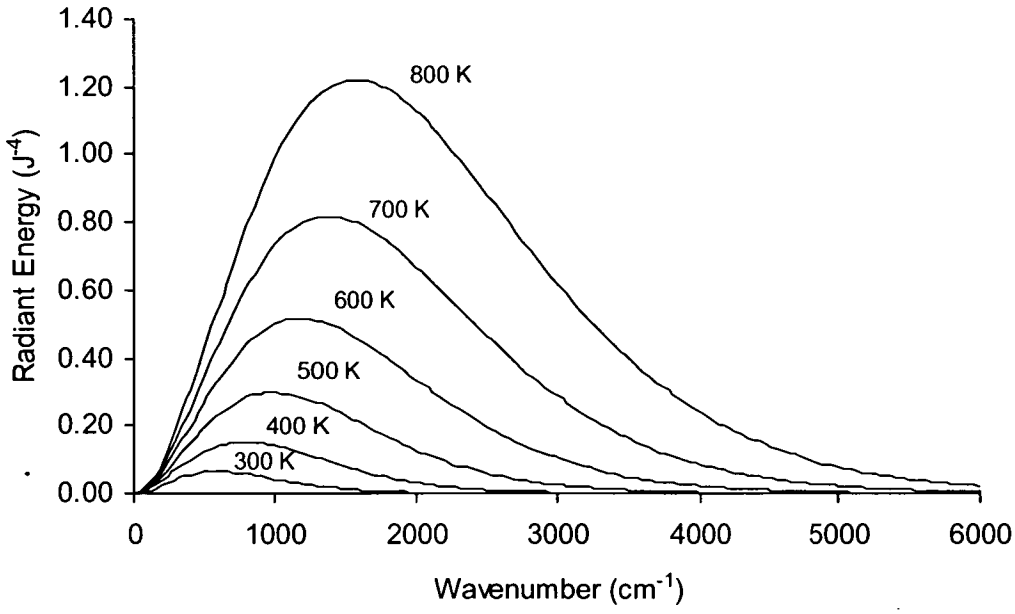


Figure 3.1 Blackbody radiation curves (Plank distributions, ρ) for various emitter temperatures

If the derivative of the Planck distribution with respect to λ is set to zero, the wavelength of the maximum in the distribution can be found at a given temperature from:

$$\frac{hc}{\lambda kT} \cdot \frac{e^{hc/\lambda kT}}{e^{hc/\lambda kT} - 1} = 5 \tag{3}$$

which, for $hc >> \lambda kT$ becomes Wien's Law:

$$\lambda_{max} T \approx \frac{hc}{5k} \tag{4}$$

Wien's Law can be used to estimate the temperature of a blackbody from its radiation curve, but the approximation made above breaks down at lower temperatures and absorbance bands due to atmospheric water and carbon dioxide obscuring the maximum in the curve.

The single beam spectrum $S(\bar{\nu},T)$ collected using an FTIR spectrometer is given by [18, 19]:

$$S(\bar{\nu},T) = R(\bar{\nu})[\varepsilon(\bar{\nu},T)H(\bar{\nu},T) + B(\bar{\nu}) + I(\bar{\nu})\rho(\bar{\nu})] \tag{5}$$

where $R(\bar{\nu})$ = instrument response function

$\varepsilon(\bar{\nu},T)$ = emittance (spectral emissivity) of sample

$H(\bar{\nu},T)$ = Planck function

$B(\bar{\nu})$ = background radiation

$I(\bar{\nu})\rho(\bar{\nu})$ = background radiation reflected from sample

$\rho(\bar{\nu})$ = reflectance of sample

$\bar{\nu}$ = frequency in wavenumbers (cm^{-1}) = $\dfrac{1}{\lambda(\text{cm})}$

T = sample temperature (kelvin)

This equation would account for the observed spectrum if the IR detector were operating at absolute zero. However, for a detector operating at ambient temperature (T_d), the observed response is:

$$S(\bar{\nu},T) - S(\bar{\nu},T_d) = R(\bar{\nu})\varepsilon(\bar{\nu})[H(\bar{\nu},T) - H(\bar{\nu},T_d)] \tag{6}$$

This assumes that the emissivity, $\varepsilon(\bar{\nu})$, is independent of temperature over the range of interest.

In order to determine the temperature of an object from its single beam spectrum, $R(\bar{\nu})$ and $\varepsilon(\bar{\nu})$ must be known or eliminated through a careful choice of spectroscopic experiments. This is one of the central problems in radiometry, because properties such as $\varepsilon(\bar{\nu})$ can only be determined if the temperature of the sample is already known. For example, a blackbody reference material at the

same temperature as the sample can be used to enable the determination of the spectral emissivity, $\varepsilon(\bar{\nu})$, so long as identical IR collection geometries are used for both [18]. A number of multi-temperature methods (involving the subtraction and ratioing of spectra taken at different sample temperatures) have been proposed to eliminate the unknown functions (ε, R, B, etc) from the spectral response of a sample, so that the temperatures can be determined using multivariate fitting techniques [10, 14]. These techniques are limited by a need to have at least one or two known temperatures to obtain a unique and reasonably accurate set of solutions for the unknown temperatures.

The approach used in the method presented here is to measure the temperature of a gas sensor at two heater voltages using the known melting points of two inorganic salts. At each of the two voltages where the temperature is determined in this way, the product $R(\bar{\nu})\varepsilon(\bar{\nu})$ can then be determined by dividing the single beam emission spectrum of the sensor (equation (6)) by the Planck function $H(\bar{\nu},T) - H(\bar{\nu},T_d)$. Thus, at any voltage where the temperature is still unknown, the sensor spectrum can be divided by $R(\bar{\nu})\varepsilon(\bar{\nu})$ to obtain a Planck function, $H(\bar{\nu},T) - H(\bar{\nu},T_d)$, for which the unknown temperature T can be determined using a least squares fitting technique.

As mentioned in the Introduction (Section 3.1), measurement of the variation with temperature (voltage) of the resistance of the heater element of the TGS8XX series sensors does not yield any useful information, because the change in resistance over the operating range of the sensor is so small (see Section *3.4.3*). However, in the case of the TGS2611 sensor, such a measurement does show significant variation of resistance with heater voltage. Once the temperature-

voltage relationship has been found using the method above, the sensor can be calibrated so that its temperature can be estimated at any voltage *without removing the cap,* simply by measuring the heater resistance. This assumes that the heater element and the sensor surface are in thermal equilibrium, which is reasonable given the construction of the sensor (see Chapter 2).

3.3 Experimental

3.3.1 Sensors

Temperature versus heater voltage measurements were conducted on two Figaro gas sensors, the TGS813 and TGS2611, which are described in Chapter 2.

In order to observe the melting points of the inorganic salts and to collect IR emission spectra from individual sensors, the sensor caps were removed (Figure 3.2(a and b)), and, in the case of the TGS2611, a hole was drilled through the base of the sensor (Figure 3.2(b)). This was done to give the spectrometer a direct 'view' of the tin oxide sensing surface, which faces the base of the sensor (see Figure 2.5, Chapter 2). One TGS813 sensing element was carefully removed from the sensor base and mounted so that the spectrometer could view only the sensing element and its connecting heater wires, (Figure 3.2(c)). In all cases, an adjustable laboratory power supply was used to supply current to the heating element, controlling the heater voltage to within 10 mV.

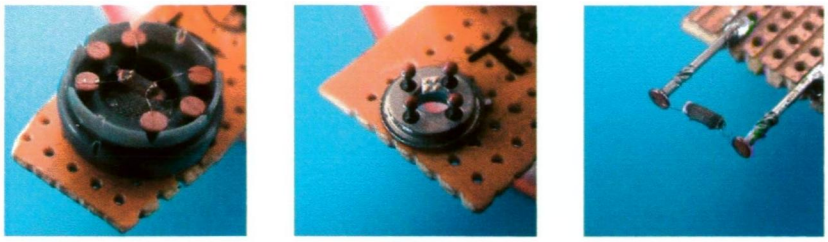


Figure 3.2 Sensors prepared for mounting in FTIR spectrometer (a) TGS813 sensing element (no cap); (b) TGS2611 sensor (no cap and hole drilled in base); (c) TGS813 sensor (no cap)

3.3.2 Melting point observations

To observe the melting points of the inorganic salts, the sensors were mounted so that the sensing surface could be examined using a 50 X dissecting microscope. Sodium chlorate (A.R. grade m.p. 248 °C) and silver chloride (A.R. grade m.p. 455 °C) were finely ground; a few grains of the fine powder were then pressed into the sensor surface in such a way as to maximise the thermal contact between the sensor surface and the powder. The sensor heater voltage was then slowly increased until the grains melted. The melting points of both salts were quite reversible, and this enabled the determination of the voltage to be repeated several times for each sensor, with the average voltage being recorded. The actual melting points of the salts (and their reversibility) were checked on bulk samples using a thermocouple (± 2 °C). To test variation between individual sensors of the same type, three TGS813 sensors were examined in this way.

3.3.3 *Collection of sensor emission spectra*

Spectra were collected using the FTIR spectrometer described in Chapter 2, operating in single beam mode. Prior to measurements being performed, the spectrometer source was disconnected from its power supply, and removed entirely from its water-cooled jacket. The gas sensor to be measured (mounted on a circuit board) was placed inside the water-cooled jacket and the water flow stopped (Figure 3.3). The spectrometer was then left for several hours to enable the water jacket to thermally equilibrate with its surroundings.

Figure 3.3 Placement of sensor inside spectrometer source water jacket

Once the spectrometer had thermally stabilised and no thermal emission was observed spectroscopically, a current was supplied to the gas sensor heater at the required voltage. The sensor temperature was allowed to stabilise for sixty seconds after the application of the current, at which point a single beam spectrum (16 scans, 8 cm^{-1} resolution, aperture open, sensitivity = 1) was collected. Upon completion of the scans, the sensor heater current was removed and thermal stabilisation of the spectrometer was allowed to occur again. This minimised thermal emission from surrounding objects (e.g. the sensor base) that had been warmed by the sensor, and prevented them from contributing to subsequent measurements. The experimental set-up required that all spectra collected were

from sensors operating in static ambient air (the spectrometer was not purged and the interferometer bearing was supplied with air rather than nitrogen).

Eleven spectra were collected in this manner, with the sensor heater voltage increased in 0.5 V increments, over the range 1.0 to 6.0 volts. Spectra were then collected with the sensor heater voltage set to the two values recorded for the melting point observations. All spectra were exported as ASCII X,Y text data files for import into Microsoft Excel 97/98 for analysis.

3.3.4 Data analysis

In Excel, the spectra obtained at the two known temperatures were divided though by the appropriate Planck function ($H(\overline{\nu},T) - H(\overline{\nu},T_d)$) to obtain the instrument response-emissivity function ($R(\overline{\nu})\varepsilon(\overline{\nu})$) spectrum. Spectra corresponding to unknown temperatures were divided through by the average of the two $R(\overline{\nu})\varepsilon(\overline{\nu})$ spectra. In each case, a Planck function $H(\overline{\nu},T) - H(\overline{\nu},T_d)$ was fitted to the resulting spectrum using a least squares routine applied through Excel Solver to determine T. The ambient (FTIR detector) temperature inside the FTIR spectrometer was found to be 298 K, and this value was used as T_d in all analyses.

3.3.5 Infrared thermometer (IRT)

An IRT was used to establish its limitations in measuring gas sensor temperatures. The model used was the Fluke 80T-IR/E extended range (32-1000 °F, 0-538 °C) infrared temperature probe, which has a minimum spot diameter of 2.5 mm at zero distance from the object being measured. The target emissivity is assumed (by the instrument) to be 0.95.

3.3.6 *Variation of heater resistance with temperature (TGS2611 sensor)*

The heater resistance was determined by measuring the voltage drop across a resistor in series with the heater, while the sensor was being operated at various heater voltages. The value of the series resistance (1 Ω) was chosen so as not to significantly alter the voltage drop across the heater.

3.4 Results and discussion

3.4.1 *Determination of* $R(\overline{v})\varepsilon(\overline{v})$

Table 3.1 gives the sensor heater voltages that correspond to the melting point temperatures of silver chloride (AgCl) and sodium chlorate ($NaClO_3$), for three different individual TGS813 sensors, and for a single TGS2611 sensor. The voltages for all three TGS813 sensors differ by no more than 20 mV at each of the two known temperatures, so it is clear that the temperature/voltage behaviour of an individual sensor can be taken as characteristic of all sensors of its type.

Sensor	*Material*	*Melting Point (°C)*	*Heater Voltage (V)*
TGS813 (a)	AgCl	455	4.91
	$NaClO_3$	248	3.06
TGS813 (b)	AgCl	455	4.90
	$NaClO_3$	248	3.04
TGS813 (c)	AgCl	455	4.90
	$NaClO_3$	248	3.06
TGS2611	AgCl	455	4.61
	$NaClO_3$	248	2.66

Table 3.1 Sensor voltages corresponding to AgCl and $NaClO_3$ melting point temperatures (note: (a), (b) and (c) are different sensors)

Figure 3.4(a) shows the instrument response function / emissivity spectra ($R(\overline{\nu})\varepsilon(\overline{\nu})$) corresponding to the two calibrated temperatures (521 K at 3.06 V and 728 K at 4.90 V) for the TGS813 sensor which had had its base removed. Ideally, the two spectra would be identical if the emissivity of the sensor did not vary significantly over the operating temperature range and if the two temperatures had been accurately determined by the melting point technique. Although the spectra in Figure 3.4(a) are not identical, very small adjustments to the two temperatures (~± 3 K) are enough to give very good agreement between the spectra, as shown in Figure 3.4(b). This justifies the assumption that the emissivity is fairly insensitive to temperature, and suggests that the numerical average of the two spectra would be very close to the 'true' $R(\overline{\nu})\varepsilon(\overline{\nu})$ function. A similar agreement was obtained for the two temperatures using the TGS813 sensor with its base still attached (Figure 3.5(a)), and for a TGS2611 sensor (Figure 3.5(b)). The temperature adjustments required to cause the spectra to overlay give an estimate of the uncertainty in the temperatures determined by this method. Thus the temperatures obtained (see section 3.4.2) for the TGS813 without base are estimated to be within ±3 K of the true temperature, while those obtained for the TGS813 and the TGS2611 with their bases on are estimated to be within ±5 K of the actual temperatures.

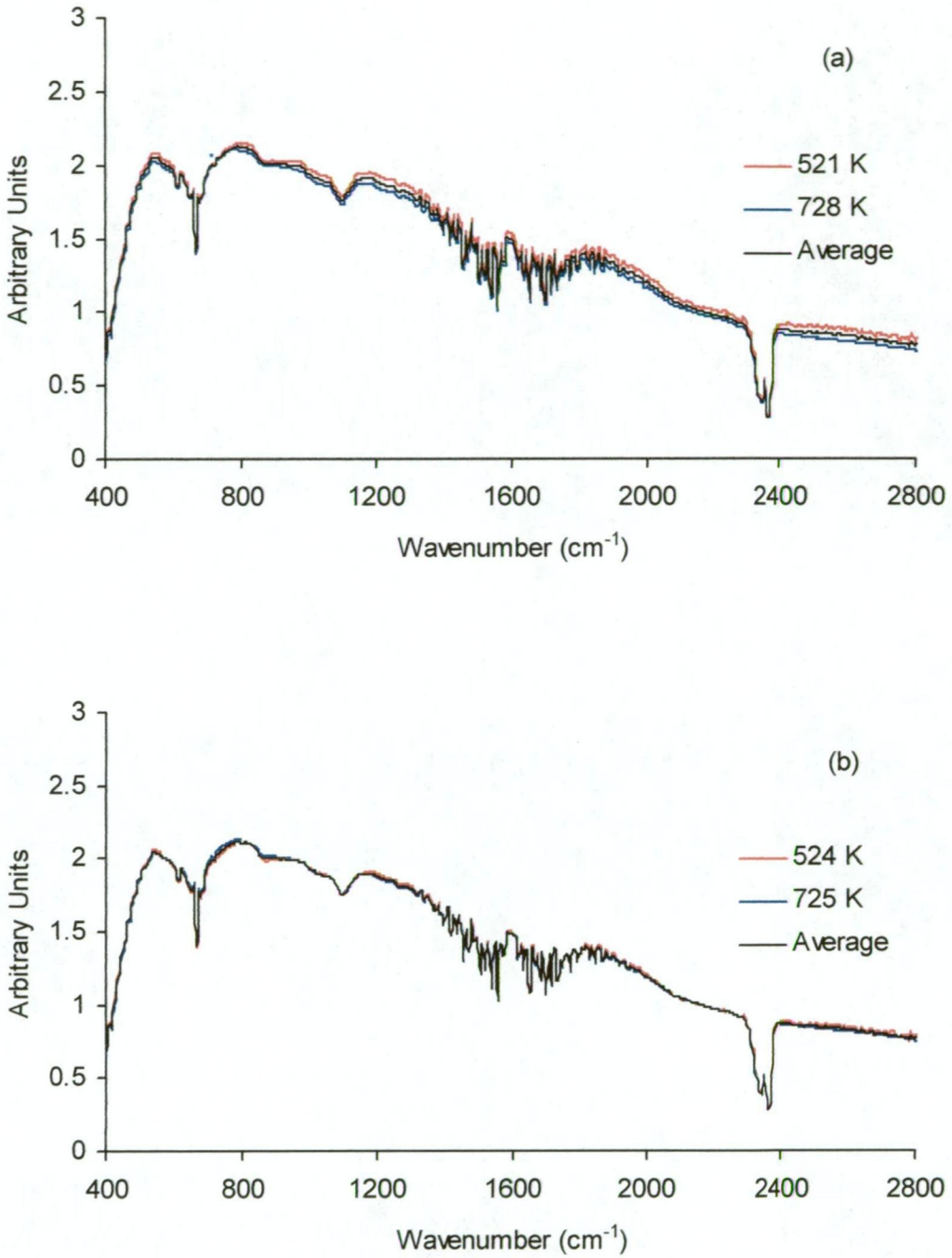


Figure 3.4 Instrument response-emissivity functions ($R(\overline{\nu})\varepsilon(\overline{\nu})$) for a TGS813 sensor (with base removed) (a) for unadjusted melting point temperatures; (b) for temperatures adjusted from those in (a)

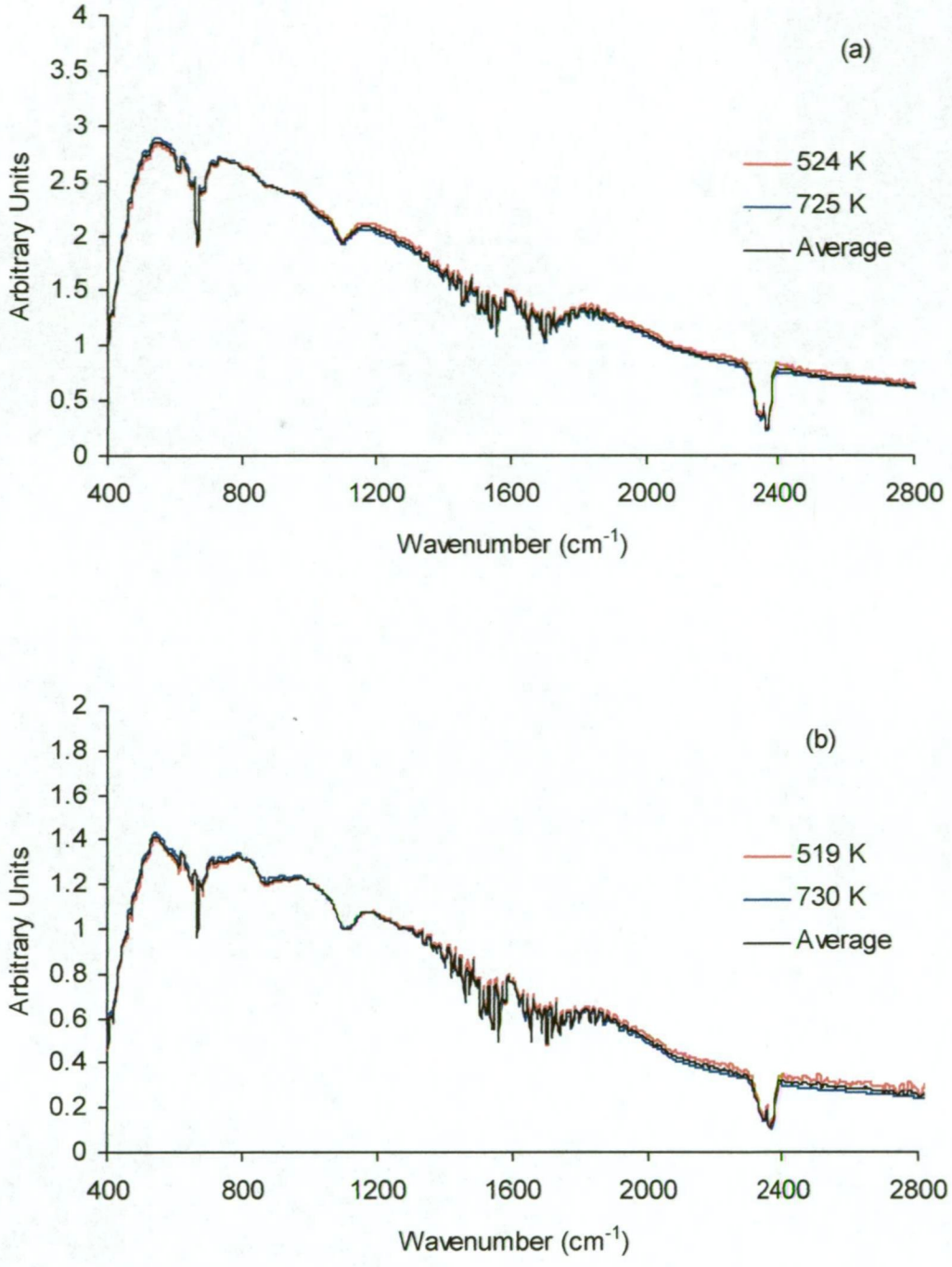


Figure 3.5 Temperature adjusted instrument response-emissivity functions ($R(\overline{\nu})\varepsilon(\overline{\nu})$) for (a) TGS813 sensor (with base) (b) TGS2611 sensor

Figure 3.6 shows selected sensor (TGS813, no base) emission spectra that have been corrected using the average $R(\overline{\nu})\varepsilon(\overline{\nu})$ function and to which the Planck function $H(\overline{\nu},T) - H(\overline{\nu},T_d)$ has been fitted in order to determine T. It can be seen that the fits are generally excellent, but decline slightly in quality at the highest voltages, presumably due to a slight temperature-dependent variation in the emissivity of the sensor surface. The fitting of results from the TGS813 sensor with base and the TGS 2611 were similarly straightforward as shown in Figures 3.7 and 3.8.

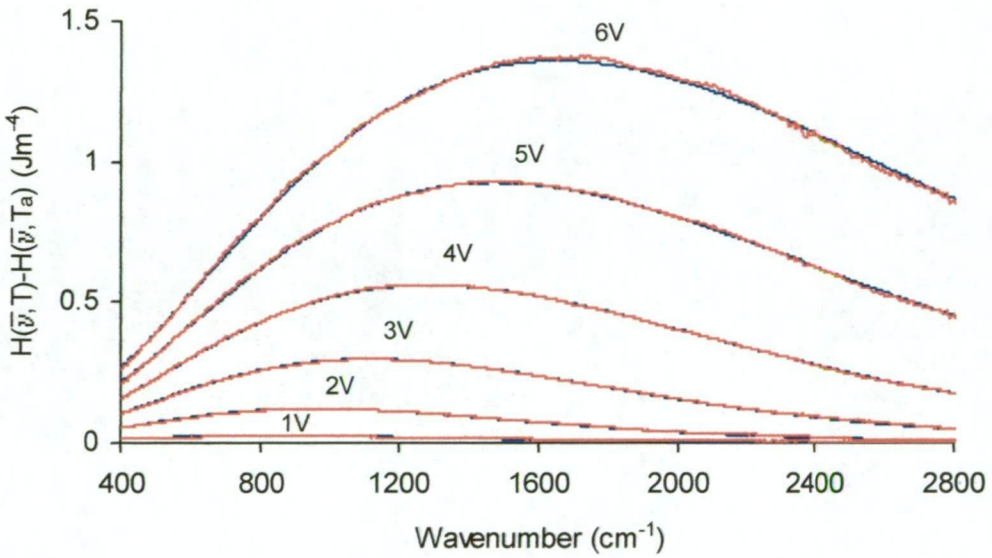


Figure 3.6 Corrected (using the average $R(\overline{\nu})\varepsilon(\overline{\nu})$ function) emission spectra for TGS813 (no base) with fitted Planck function, $H(\overline{\nu},T) - H(\overline{\nu},T_d)$

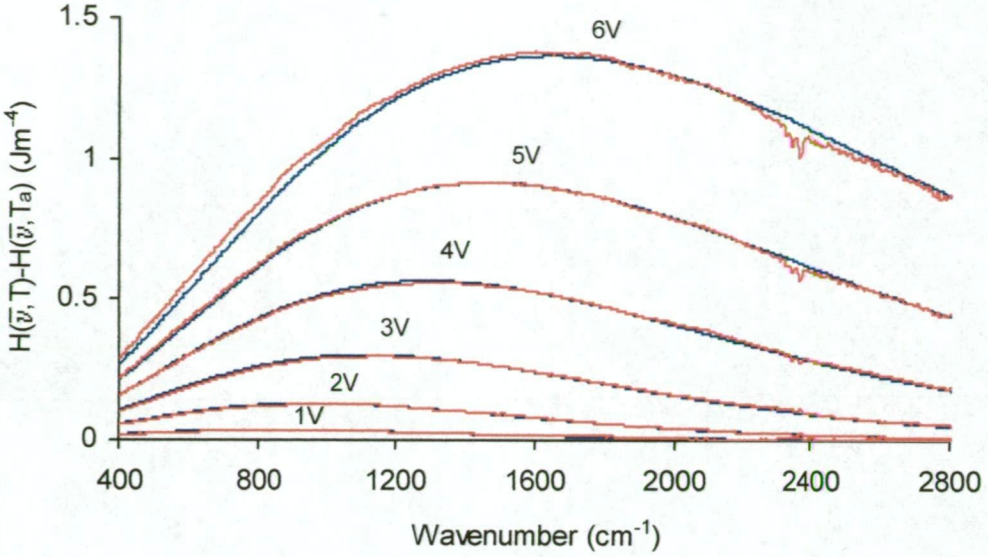


Figure 3.7 Corrected (using the average $R(\overline{\nu})\varepsilon(\overline{\nu})$ function) emission spectra for TGS813 (with base) with fitted Planck function, $H(\overline{\nu},T) - H(\overline{\nu},T_d)$

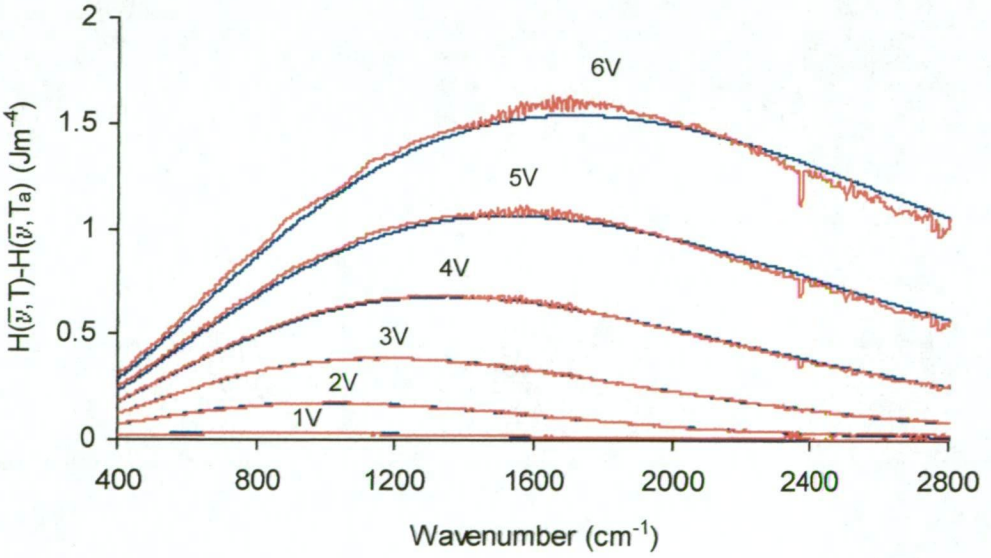


Figure 3.8 Corrected (using the average $R(\overline{\nu})\varepsilon(\overline{\nu})$ function) emission spectra for TGS2611 with fitted Planck function, $H(\overline{\nu},T) - H(\overline{\nu},T_d)$

3.4.2 *Temperature measurements and heat loss model*

Figure 3.9, 3.10 and 3.11 are plots of the temperatures determined for the TGS813 sensor (no base, base) and TGS2611 sensor operating at all eleven of the selected heater voltages. The plots appear remarkably linear over the voltage range plotted, and the equations of fitted lines are given. Closer examination of Figure 3.9 reveals that the actual data is slightly sigmoidal in shape since it is obliged to have an intercept of 298 K (the ambient temperature) at 0 V. Observations of sensor behaviour below 1 V indicate that no appreciable heating occurs below 0.5 V, thus the sensor is at ambient temperature at 0.5 V. It can be seen from the data point at 1.0 V that the temperature values are starting to curve so as to approach ambient temperature at 0.5 V. All of the sensors measured in this work appear to approach ambient temperature at about 0.5 V as the heater voltage is decreased.

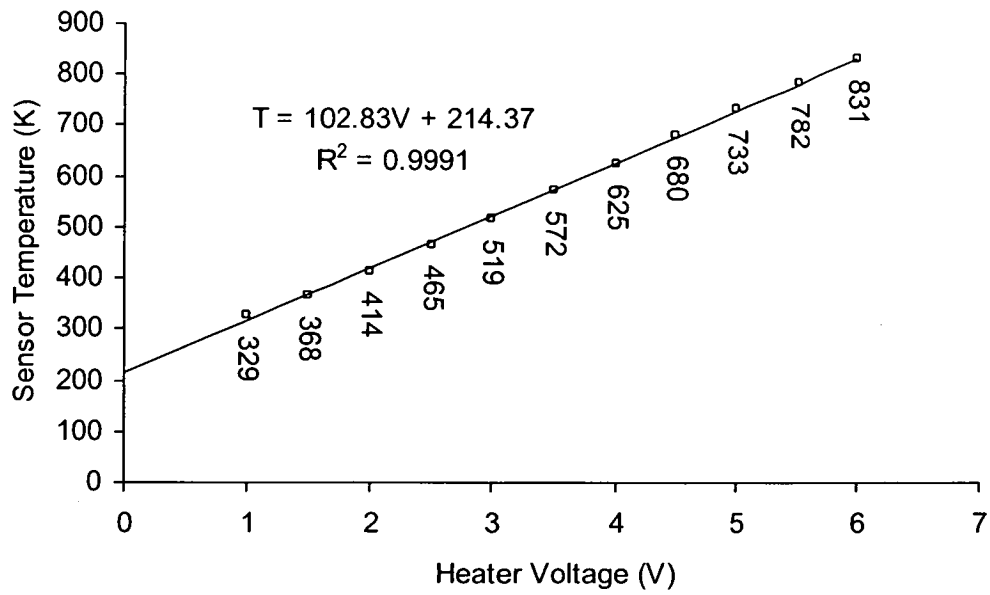


Figure 3.9 Plot of TGS813 (no base) temperature at various heater voltages

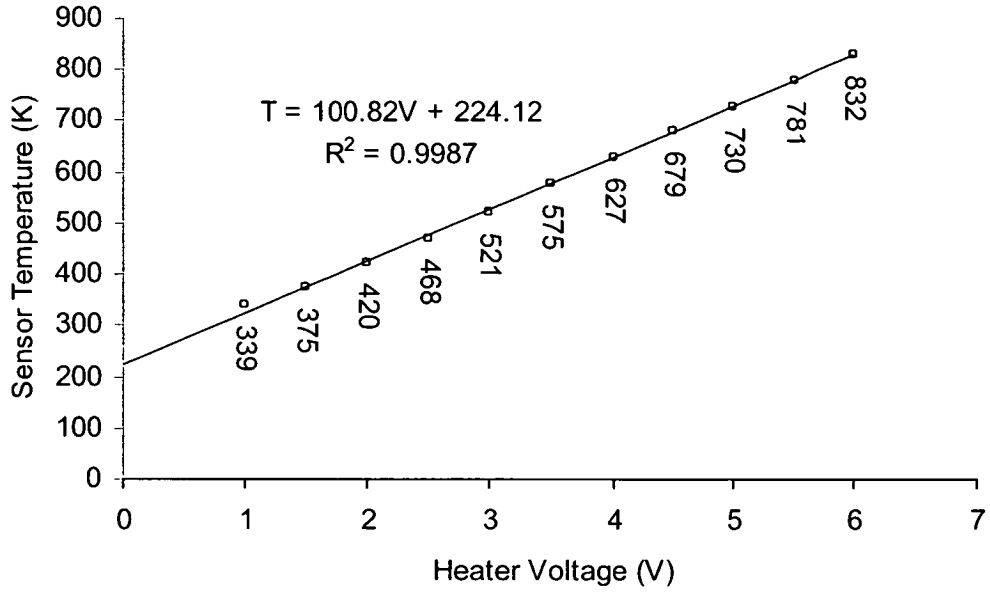


Figure 3.10 Plot of TGS813 (with base) temperature at various heater voltages

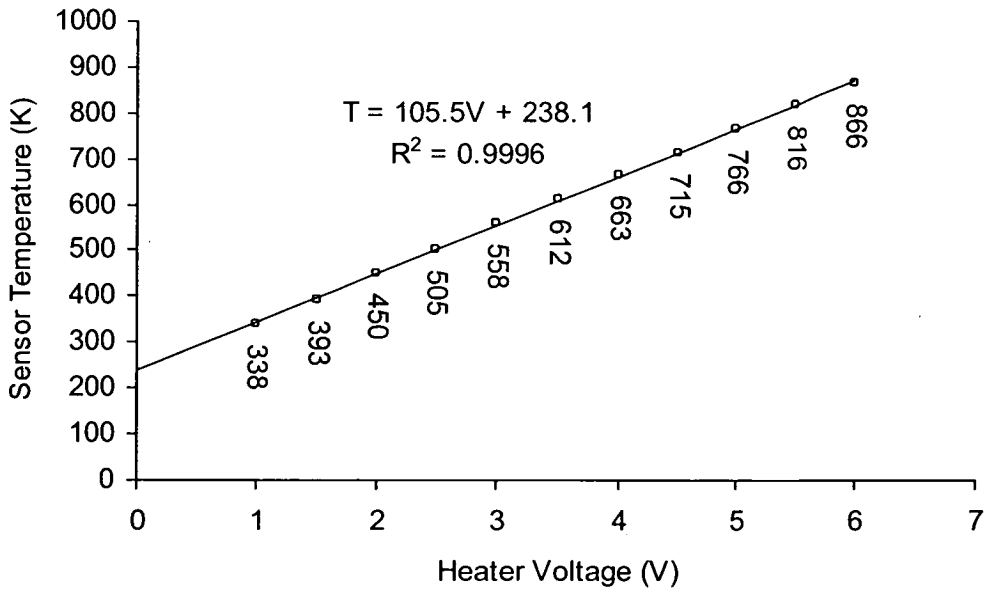


Figure 3.11 Plot of TGS2611 temperature at various heater voltages

A naïve treatment of the power balance (electrical power input and heat loss) for a gas sensor suggests that if the only mechanism for heat loss from the sensor is radiative, the following relationship can be derived, in which the sensor is considered to be a greybody radiator, of total emissivity ε and surface area A:

$$\frac{V^2}{R} = \varepsilon\sigma A(T^4 - T_a^4) \tag{7}$$

In this equation, V is the applied heater voltage, R is the resistance of the heater element, T is the temperature of the sensor, T_a is the ambient temperature and $\sigma = \frac{ac}{4}$ (see Equation (1)). Thus a plot of T^4 versus V^2 (rather than T versus V) would be expected to be linear [20]. That the former does not hold is indicative of the importance of other heat loss mechanisms (conduction, convection). A pseudo-linear relationship ($T = (104V + 196 \pm 5)$ K) between temperature and heater voltage has also been found by Clifford [11] for the TGS812 sensor, which has the same construction and power consumption characteristics as the TGS813. If one can adopt Clifford's model for heat loss from the sensor, which includes a linear term for the conduction of heat through the ambient atmosphere around the sensor, the following equation is obtained:

$$\frac{V^2}{\sigma\varepsilon AR} = (T^4 - T_a^4) + \frac{k}{\sigma\varepsilon l}(T - T_a) \tag{8}$$

Here, k ($\mathrm{Wm^{-1}K^{-1}}$) is the thermal conductivity of the ambient gas and l (m) is the distance from the sensor over which the temperature of that gas drops to the ambient temperature. This implies that a plot of $(T^4 + \frac{k}{\sigma\varepsilon l}T)$ versus V^2 should be

linear, with a slope of $\frac{1}{\sigma\varepsilon AR}$ and an intercept of $T_a^4 + \frac{k}{\sigma\varepsilon l}T_a$ (T will be equal to T_a when $V = 0$). This provides a means of estimating the emissivity ε if the ratio $\frac{k}{l}$ is known or if the effective surface area of the sensor (A) is known. The ratio $\frac{k}{l}$ was found by Clifford to be approximately 36.12 $Wm^{-2}K^{-1}$ for a TGS812 sensor (with cover on) in air. If this value is assumed to be similar for the TGS813, an estimate of the emissivity of the sensor surface can be obtained through regression analysis. Figure 3.12, which has been plotted using temperature measurements for a TGS813 sensor with its base on (to match most closely the situation modelled by Clifford), shows the linear plot obtained when the emissivity has a value of 0.84. This is a realistic value for ε, suggesting that this is a reasonable model for heat loss by the sensor (although the effects of convection and heat conduction through the sensor wires have been ignored). However, since $\frac{k}{l}$ and ε are really both unknowns and cannot be determined independently of each other, this estimate of ε should be seen as approximate at best. For example, when similar fitting is performed on data from the TGS813 *without* its base, a value of 0.91 is obtained for ε, using the same (but now less valid) estimate for $\frac{k}{l}$. Plotting the TGS813 data in this way yields an apparent ambient temperature (T_a) which is slightly higher than the actual temperature in the spectrometer, and this also points to limitations in this approach to modelling sensor heat loss. It should be emphasised here that the limitations in this model do not compromise the radiometric temperature measurements made in this work as they do not need to take conduction or convection effects into account.

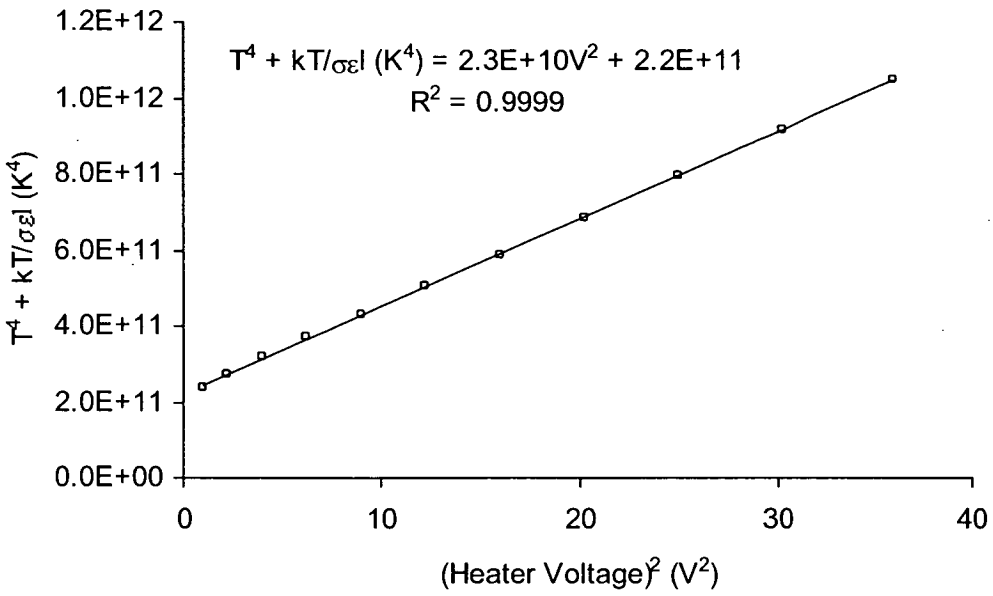


Figure 3.12 Demonstration of heat loss model (Eq. (8)) for a TGS813 (with base) sensor

Table 3.2 compares the voltage/temperature data obtained in the present work with all such data for Figaro gas sensors found in literature searches. It should be noted that all of the literature data in Table 3.2 is from work on sensors used in static atmosphere (non-flow) conditions. The results obtained by Clifford [11] for the TGS812 differ by only 10-30 K over the operating voltage range from the measurements on the TGS813 in this work. As mentioned above, the two sensors have the same construction and may differ only slightly in the composition of the sensing material, so they would be expected to exhibit very similar voltage/temperature behaviour. Since the TGS812 is no longer available, a direct comparison is not possible, but the present work represents the first (albeit partial) validation of the method of Clifford. In contrast, the estimate of 573-673 K made by Sears et al [12] for the TGS812 operating at 5.0 V differs greatly from Clifford's value of 716 K for the same sensor and the measurement of 730 K (for

the TGS813 in this work). Interestingly, the value (~700 K) given by the manufacturer, Figaro, for the TGS813 at 5.0 V is quite close to the radiometric TGS813 value of this work (and Clifford's, 716), but the method by which it was obtained is unknown.

Heater voltage (V)	Sensor temperature (K)							
	TGS813				TGS813 (no base)	TGS812		TGS2611
	This work ±5 K	Figaro	IRT (this work)	IRT Ref. [6]	This work ±3 K	Ref. [12]	Ref. [11]	This work ±5 K
0.5			300					
1.0	339		322		329		300	338
1.5	375		344		368		352	393
2.0	420		383	490	414		404	450
2.5	468		419		465		456	505
3.0	521		461		519		508	558
3.5	575		505		572		560	612
4.0	627		544		625		612	663
4.5	679		580		680		664	715
5.0	730	~700	616	630	733	573-673	716	766
5.5	781		652		782		768	816
6.0	832		686		831		820	866

Table 3.2 Comparison of voltage/temperature measurements for Figaro sensors operated in air, in static atmosphere (non-flow) conditions, with no analyte gas present

From literature searches, the only other temperature estimates for operating voltages of the Figaro TGS813 are those reported by Nakata et al [6], who used an IRT. Table 3.2 shows that their value at 2.0 V is 70 K higher than the radiometric value reported here, while their temperature estimate at 5.0 V is 100 K lower. Since the measurement procedure employed by Nakata et al is not specified, it can only be speculated that the discrepancies are due to the limitations identified in

Section 3.1. It should be noted, however, that all of these limitations suggest that an IRT would underestimate, rather than overestimate the temperature at every heater voltage. When temperature measurements were performed using the Fluke IRT, the temperature value obtained at 5.0 V was roughly similar (within 15 K) to that obtained by Nakata et al, but the temperature measured at 2.0 V was closer to the TGS813 radiometric value than to Nakata's IRT value (see Table 3.2). It is worth noting that the IRT temperature data is also quite linear when plotted against heater voltage, but with a significantly smaller slope, which illustrates the general underestimation of the sensor temperature by this instrument.

The temperature estimates for the TGS2611 sensor given in Table 3.2 are the first reported for a semiconductor sensor of this size and construction. Although the temperature at 1.0 V is very similar to that measured for the TGS813, from 2.0 V onwards the TGS2611 runs at about 36 K higher than the TGS813.

3.4.3 Calibration of temperature-resistance relationship in TGS26XX sensors

As mentioned in Section 3.1, the temperature coefficient of typical heater wire in TGS8xx series sensors is too low to give precisely measurable changes in resistance over the operational voltage range. The resistance of a TGS813 sensor heater wire was measured and found to be effectively constant at 32 Ω over the entire operating voltage range (0-6.0 V). While there has been uncertainly in some quarters as to the composition of the heater wire in TGS8XX sensors (it is not specified in any Figaro data sheets), this observation seems to rule out platinum (temperature coefficient = 0.003927 K^{-1}) as a possibility. This question was further investigated by conducting atomic absorption spectroscopy (AAS) and

other chemical tests on a solution made up of TGS813 heater wires dissolved in aqua regia. Qualitative tests for platinum gave negative results, while the AAS measurement indicated that the heater wire is composed mainly of iron and chromium, suggesting that the wire is a type of stainless steel. These results were confirmed by EPMA which estimated the composition of the wire to be 77 % iron and 23 % chromium.

The TGS2611 series sensors have a platinum heating element, which does not have a constant resistance over the operating voltage range [1]. Instead, the resistance of the platinum element was found to vary linearly with voltage, and hence with temperature, giving the effectively linear relationship between temperature and voltage reported here. (This makes the assumption that the heating element and the sensing element are at the same temperature. This assumption is reasonable given the close contact between these two layers, which are printed onto opposite sides of a very thin alumina substrate.) Figure 3.13 shows the temperature-resistance relationships for a TGS2611 sensor operating with and without its cover. They are plotted on different resistance axes here for convenience as they almost overlay. This indicates that the sensor's operating temperature under non-flow (static atmosphere) conditions is only slightly affected by the presence of its cover. This may seem surprising at first, but the cover is much larger than the actual sensor, and is metallic, in contrast to the plastic base and cover surrounding the TGS8XX sensors. Being metallic, the cover has a very low emissivity and thus conducts heat very efficiently to the external environment.

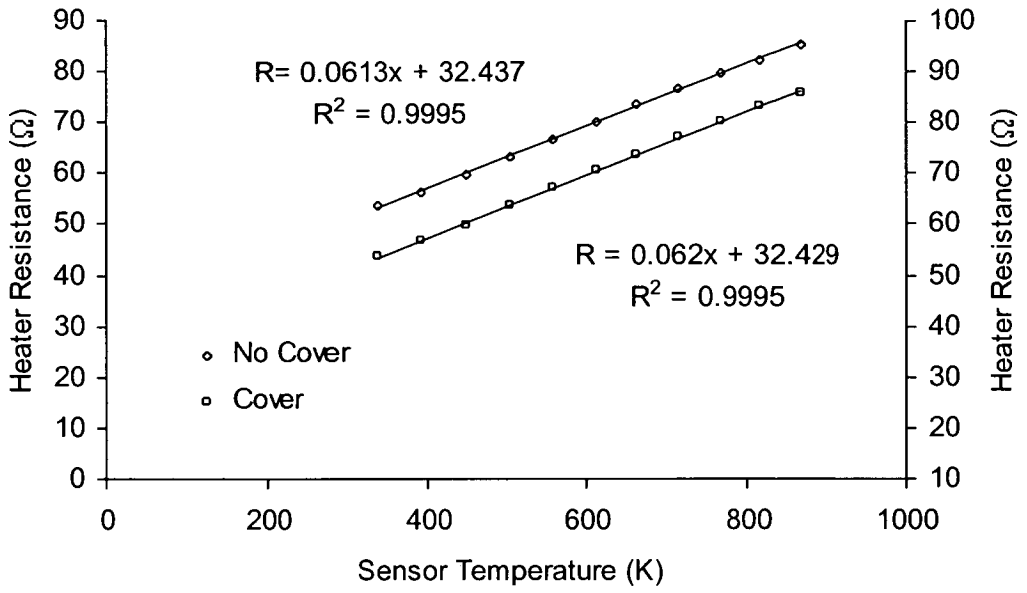


Figure 3.13 Sensor temperature versus heater resistance plots for TGS2611 sensor with cover on (upper plot, left axis) and cover removed (lower plot, right axis), plots are offset for clarity

These results suggest that the method discussed in Section 3.1 for determining a sensor's temperature by measuring its heater resistance in an oven would actually be feasible for TGS26XX sensors where it would have failed for the TGS8XX sensors. However there would be some complications to be overcome (e.g. variation in the resistance with temperature of connecting wires). This could possibly be done using a Kelvin measurement (four wires), but this could give rise to parasitic Seebeck voltages due to the pin materials being dissimilar to the connecting wires. Since all of the Figaro TGS26XX series sensors have the same platinum heating element, it seems probable that they would all exhibit linear temperature-resistance relationships, and thus would need only one calibrating

temperature measurement (apart from ambient) to enable temperature determination at all applied heater voltages.

It should be stressed that in the absence of analyte gases in these temperature determinations, the effects of Joule heating of the sensor surface by analyte combustion reactions have been ignored. However, from resistance measurements on the TGS2611, and from those of Heilig and co-workers [22], a temperature change of about 3 K can be measured when a Pd-doped tin oxide sensor operating at about 550 K is exposed to 1000 ppm ethanol in air. In the case of the TGS8XX sensors, this temperature change must be added to the uncertainty in the temperature measurements reported in this work, when an analyte gas is present. However, the resulting uncertainty is still relatively small compared with that suffered by IRT measurements. In the case of TGS26XX series sensors, the ability (after calibration) to measure the sensing surface temperature resistively at all times means that temperature measurements can be made as easily in the presence of an analyte gas as in pure air, even under flow conditions. This is the strategy used by Heilig and co-workers [22] for sensors of their own construction, with platinum resistive heating elements.

3.5 Conclusion

Accurate knowledge of sensor temperatures is essential for research into the underlying response mechanisms of metal oxide gas sensors. This work shows that it is relatively simple to determine the surface temperature of small commercial semiconductor gas sensors using radiometric methods. The results reported here represent the best estimate to date of the non-flow operating temperatures of these sensors, and demonstrate the inadequacy of infrared thermometers for measuring the temperature of small objects of unknown emissivity.

References

[1] Figaro Engineering Inc., Technical information on usage of TGS sensors for toxic and explosive gas leak detectors, http://www.figarosensor.com/ (1996).

[2] P.T. Moseley, D.E. Williams, Oxygen surface species on semiconducting oxides, in P.T. Moseley, J.O. W. Norris and D.E. Williams (eds.), *Techniques and Mechanisms in Gas Sensing* (1991), Adam Hilger, Bristol, pp. 46-60.

[3] N. Yamazoe, N. Miura, Some basic aspects of semiconductor gas sensors, in S. Yamauchi (ed.), *Chemical Sensor Technology*, Vol. 4, (1992), Kodansha, Tokyo, pp. 19-42.

[4] A.P. Lee, B.J. Reedy, Temperature modulation in semiconductor gas sensing, *Sens. Actuators B* **60** (2000), 35-42.

[5] Y. Hiranaka, T. Abe, H. Murata, Gas-dependent response in the temperature transient of SnO_2 gas sensors, *Sens. Actuators B* **9** (1992), 177-182.

[6] S. Nakata, S. Akakabe, M. Nakasuji, K. Yoshikawa, Gas sensing based on a nonlinear response: discrimination between hydrocarbons and quantification of individual components in a gas mixture, *Anal. Chem.* **68** (1996), 2067-2072.

[7] A. Heilig, N. Barsan, U. Weimar, M. Schweizer-Berberich, J.W. Gardner, W. Göpel, Gas identification by modulating temperatures of SnO_2-based thick film sensors, *Sens. Actuators B* **43** (1997), 45-51.

[8] L. Michalski, K. Eckersdorf, J. McGhee, in *Temperature measurement* (1991), Wiley, Chichester, pp. 317-350.

[9] M. Ballico, T. Jones, Novel experimental technique for measuring high temperature spectral emissivities, *Appl. Spectrosc.* **49** (1995), 335-340.

[10] S. Clausen, A. Morgenstjerne, O. Rathmann, Measurement of surface temperature and emissivity by a multitemperature method for Fourier-transform infrared spectrometers, *Appl. Opt.* **35** (1996), 5683-5691.

[11] P.K. Clifford, Mechanisms of gas detection by metal oxide surfaces, *Ph.D. Dissertation*, Carnegie-Mellon University (1981).

[12] W.M. Sears, K. Colbow, F. Consadori, General characteristics of thermally cycled tin oxide gas sensors, *Semicond. Sci. Tech.* **4** (1989), 351-359.

[13] J. Lübben, B. Schrader, Chemical composition analysis and temperature determination of optically levitated single aerosol particles by means of a compact Raman spectrometer, *J. Mol. Struct.* **410-411** (1997), 543-546.

[14] J. Markham, P. Best, P. Solomon, Spectroscopic method for measuring surface temperature that is independent of material emissivity, surrounding radiation sources and instrument calibration, *Appl. Spectrosc.* **48** (1994), 265-270.

[15] S. Clausen, L. Sørensen, Measurement of single moving particle temperatures with an FT-IR spectrometer, *Appl. Spectrosc.* **50** (1996), 1103-1111.

[16] J. Markham, K. Kinsella, R. Carangelo, C. Brouillette, M. Carangelo, P. Best, P. Solomon, Bench top Fourier transform infrared based instrument for simultaneously measuring surface spectral emittance and temperature, *Rev. Sci. Instrum.* **64** (1993), 2515-2522.

[17] P. Morrison, P. Solomon, D. Hamblen, Method and apparatus for temperature determination, U.S. Pat. 4985858 (1991).

[18] D. Chase, The sensitivity and limitations of condensed phase infrared emission spectroscopy, *Appl. Spectrosc.* **35** (1981), 77-81.

[19] P. Griffiths, J. de Haseth, in *Fourier Transform Infrared Spectroscopy* (1986), Wiley, New York, pp. 202-205.

[20] S. Wlodek, K. Colbow, F. Consadori, Signal-shape analysis of a thermally cycled tin-oxide gas sensor, *Sens. Actuators B* **3** (1991), 63-68.

[21] Figaro Engineering Inc., Technical information for volatile organic compound (VOC) sensors, http://www.figarosensor.com/ (1997).

[22] A. Heilig, N. Barsan, U. Weimar, W. Göpel, Selectivity enhancement of SnO_2 gas sensors: simultaneous monitoring of resistances and temperatures, *Sens. Actuators B* **58** (1999), 302-309.

Chapter 4

Oxygen response studies

4.1 Introduction

Since the development of the first metal oxide gas sensors by Taguchi in the late 1960s, no satisfactory model has been put forward to explain the non-linear response of these devices to analyte gases. In spite of intensive research and development in this area, most models for sensor response are empirical at best. This can largely be explained by the complexity of the processes involved and by the fact that expertise in a number of diverse fields (surface chemistry, solid state physics, materials science, spectroscopy) is required for complete understanding of these processes. As pointed out by Lampe and co-authors [1] in a recent survey of the field, most existing models can only be applied to a particular temperature range of a particular metal oxide with a specific analyte gas. Even for tin (IV) oxide (SnO_2) based sensors, a bewildering array of different models have been proposed for just a few analyte gases [2-7]. Since SnO_2 is the predominant material used in commercial sensors, and since it has some peculiar properties when compared with other metal oxides [8], a more complete understanding of the sensing properties of this material alone is desirable.

It is universally agreed that the response of tin oxide gas sensors is based on the removal from the oxide surface by an analyte gas of adsorbed oxygen species which, when present, increase the resistance of the surface by withdrawing electron density. The surface-catalysed combustion of analyte gas thus decreases the resistance of the surface, but to an extent that is non-linearly related to the analyte concentration. The more controversial issues that have been examined by various workers are: (i) the speciation of adsorbed oxygen at different temperatures; (ii) the role of 'dopants' such as palladium in oxygen and analyte

adsorption; (iii) the type of analyte adsorption (if any) involved, and its effects; (iv) surface versus bulk conduction through polycrystalline SnO_2; and most importantly, (v) the relationship between surface coverage of oxygen and the resistance (or conductivity) of the material.

The bulk of the evidence assembled to date provides a fairly consistent picture of the dependence of surface oxygen speciation on temperature, although publications in the gas sensing field which are at odds with the generally accepted scheme still appear. This is probably because the variation in properties of different oxides has led to dangerous generalisations about these compounds. For this reason this discussion will be confined to tin oxide only. Care must also be taken to distinguish between pure and 'doped' SnO_2, and to specify sample preparation methods. Firstly, there is agreement that the oxygen speciation moves with increasing surface temperature from diatomic (O_2, O_2^- or O_2^{2-}) to monatomic (O^- or O^{2-}) [9-11]. EPR measurements on pure tin oxide by Chang [9] indicated that the transition temperature between O_2^- and O^- occurs at around 150 °C. Chang also observed the subsequent reduction of the O^- signal between 200 and 280 °C. This was attributed to the formation of O^{2-}, which has no EPR signal, but could also be explained by the desorption of O^- with increasing temperature.

It is difficult to correlate these EPR results with the thermal desorption data of Yamazoe [10] because of the differing nature of the techniques, but the latter indicates that the predominant species below 150 °C is diatomic (O_2 or O_2^-), while above that temperature, it is monatomic (O^- or O^{2-}). Mizokawa and Nakamura [12] also reported the desorption of O_2^- from SnO_2 at about 150 °C. Adsorption rate equation simulations [13] predict the temperature-dependent transition from

O_2^- to O^- at around 170 °C. In a more recent XPS study, Nagasawa et al [14] suggest that O^- and O_2^{2-} are desorbed above 200 °C, while O_2^- is desorbed above 400 °C. These results are seemingly at odds with the earlier findings. One theoretical study by the same group [15] predicts that although the catalytically active species on the SnO_2 surface are O^- and O_2^-, these species are in low concentration because they readily form the more thermodynamically favoured O^{2-} and O_2^{2-}. Another theoretical study by Yamaguchi et al [16] finds that chemisorbed O_2 can either take the form of a peroxo species (O_2^{2-}) in side-on adsorption or a superoxo species (O_2^-) in end-on adsorption. This study also predicts a low dissociation barrier for adsorbed O_2^{2-} to give O^- or O^{2-}. However, neither studies make predictions about the dependence of oxygen speciation on temperature.

Another possibility for the speciation of adsorbed oxygen is the hydroxyl group (OH^-). This can be formed from the dissociative adsorption of water on the oxide surface ($H_2O + O^{2-}_{lattice} \rightarrow 2OH^-$) or from the protonation of adsorbed O^- or O^{2-}, where the protons are derived from molecules such as dissociatively adsorbed hydrocarbons. In a summary of literature data on O_2 and H_2O adsorption and desorption from tin oxide gel-based pressed disks, Moseley et al [11] conclude that for unsintered, undoped samples, OH^- is a major conductance-modulating surface species, whose removal at higher temperatures allows an increase in the surface concentrations of O_2, O^- and O^{2-}. However, for sintered samples, conduction control is attributed primarily to O_2^- below 550 K, and to O^{2-} above this temperature. These conclusions are more relevant to sintered thick film

commercial gas sensors, although the role, if any, of hydroxyl groups in gas sensor response is not clear.

To increase sensitivity, tin oxide sensors are often 'doped' with small quantities of noble metals such as palladium. One effect of palladium (which is really dispersed on the tin oxide particles [17]) is to lower the temperature at which maximum sensitivity to an analyte occurs. This is probably because it lowers the activation energy of oxygen (and possibly analyte) dissociation on the tin oxide surface, and thus shifts the temperature profile of oxygen speciation to lower values. Dissociated oxygen species (e.g. O^-) originating at palladium particles are thought to 'spill over' onto the tin oxide surface [18]. This may also be the case with analyte molecules for which the initial adsorption onto palladium (rather than tin oxide) is favoured.

Also fundamental to an understanding of the mechanism of tin oxide gas sensitivity is the knowledge of the interaction of analyte gases with the sensor surface, and the dependence of this interaction on temperature. A given analyte gas may adsorb (dissociatively or non-dissociatively) onto the surface, and then react with adsorbed oxygen species (Langmuir-Hinshelwood mechanism) or it may react with adsorbed oxygen simply by colliding with it on the surface (Rideal mechanism). The problem here is that a given analyte may participate in different oxidation mechanisms at different temperatures. Apart from the purely mechanistic aspects of analyte adsorption, its direct effect on the conductivity of the gas sensor must also be considered – does the analyte gas donate or withdraw electron density to/from the SnO_2 surface, and if so, at what surface coverage does this effect become significant relative to the contribution by adsorbed oxygen?

Several models for the conduction processes in metal oxides have been proposed. It is of great importance to distinguish between these models on the basis of the temperature range of interest and the specific metal oxide under examination. In their review, Lampe et al [1] point out that at 'high temperatures' (> 900 °C), electrical conduction is based on the existence of bulk oxygen defects, while at lower temperatures, the conductivity is only influenced by processes on the surface (i.e. reactions of chemisorbed species). Typical commercial tin oxide gas sensors are operated at a maximum temperature of about 500 °C (see Chapter 3), under which conditions they exhibit almost metallic behaviour when adsorbed oxygen species are removed from the surface. It would thus be desirable to establish the relationship, at a given temperature, between the response of a gas sensor (measured as resistance or conductivity) and the fractional surface coverage of oxygen (θ_O or θ_{O_2}) on the sensor surface.

Apparent power law relationships between sensor conductivity and the concentrations of many analyte gases, as well as oxygen, have made the empirical Freundlich isotherm ($\theta = kp^n$) a popular choice as the basis for such a relationship [2,3,7,19,20], but these models cannot take into account the different speciation of oxygen, or competition between oxygen and analyte gases for adsorption sites. Langmuir-type isotherms, however, can be derived from a simple set of assumptions, and can be modified to take both of these phenomena into account. They have often been used in attempts to model sensor behaviour [4-6]. In this chapter, a model for tin oxide gas sensor response across a wide range of temperatures is presented, based on Langmuir-type isotherms and observations of the relationship between sensor resistance and oxygen concentration.

4.2 Theory - Langmuir adsorption

Given that the oxide surface has N adsorption sites, and assuming:

(i) that all sites are equivalent, and

(ii) that adsorption to a given site is independent of the occupation of neighbouring sites,

then the following expressions can be used to describe the rate of adsorption and desorption of oxygen:

$$\frac{d\theta}{dt} = k_a p_{O_2} N(1-\theta)^x \quad \text{(adsorption)} \tag{1a}$$

$$\frac{d\theta}{dt} = k_d N\theta^x \quad \text{(desorption)} \tag{1b}$$

where k_a and k_d are the rate constants for adsorption and desorption respectively, θ is the fractional coverage of adsorbed oxygen, N is the number of sites, p_{O_2} is the pressure of oxygen and $x = 1$ for non-dissociative adsorption or ½ for dissociative adsorption. If it is assumed that an equilibrium exists between gaseous oxygen and adsorbed oxygen then the two rate equations describing adsorption and desorption can be equated to give:

$$k_a p_{O_2} (N(1-\theta))^x = k_d (N\theta)^x \tag{2}$$

Rearranging both these equations yields the familiar Langmuir isotherm expression:

$$\theta_O = \frac{(Kp_{O_2})^x}{1+(Kp_{O_2})^x} \tag{3}$$

where the adsorption coefficient K $(= \frac{k_a}{k_d})$ is a ratio of Arrenhius-like rate constants, and behaves like an equilibrium constant for adsorption. Thus, its variation with temperature can be expressed as follows:

$$K = K_0 \exp\left(\frac{-\Delta H}{RT}\right) \tag{4}$$

where K_0 is independent of temperature and ΔH is the enthalpy change on adsorption which is often assumed to be independent of surface coverage [21]. Since most adsorption processes are exothermic, K generally decreases with increasing temperature, and so less gas is adsorbed to a given surface at higher temperatures.

4.3 Experimental

4.3.1 Sensor response versus oxygen pressure studies

The response of a sensor to static oxygen pressures was investigated using the Figaro TGS2611 sensor described in Chapter 2. All sensors were preheated in clean air for a period of one week prior to use. The oxygen pressure experiments were conducted using the custom gas manifold system (Chapter 2). Simultaneous measurements of oxygen pressure and sensor resistance were performed using the data acquisition system with the associated logging package software described in Chapter 2.

At the beginning of each experiment, the sensor was heated to its maximum normal operating temperature (~493 °C at 5.0 V); and the flask was evacuated and then isolated from the manifold. A small quantity of hexane was admitted to the

flask via the injection port to remove any remaining surface-adsorbed oxygen from the sensor. The vacuum was then re-applied for one minute to remove any remaining hexane and reaction products. The sensor temperature was then set to approximately 100 °C and the sensor resistance and flask pressure were sampled every 30 seconds while high purity oxygen was admitted to the flask at 0.3 mL/min through a 150 mm GC needle until the oxygen pressure was 0.3 atm. The sensor response was then allowed to stabilise for a period of one hour prior to the procedure being repeated with the sensor operated at maximum temperature (~493 °C). After each scan, resistance versus pressure data were transferred via text file to Excel for analysis.

4.3.2 Dynamic oxygen adsorption studies

The time-dependent response of a sensor to the surface adsorption of oxygen was studied using a modified TGS2611 sensor. The sensor cover was carefully removed and a hole was machined in the sensor base (in the TGS2611, the sensing surface material faces the base). The sensor was mounted on a circuit board through which passed a 20 × 1.5 mm diameter brass tube. The end of the tube was positioned approximately 0.2 mm from the sensing surface so that any gas exiting the tube would completely envelop the sensor. The sensor cover was then re-installed to prevent ambient air movements affecting sensor response. A modified Swagelok T-fitting was mounted on the other end of the tube with septa fitted to the two branches.

With the sensor operating at its maximum normal operating temperature (~493 °C at 5.0 V, see Chapter 3), a pure air flow of 10 mL/min was admitted to the sensor via the T-branch perpendicular to the brass tube. In order to remove surface-

adsorbed oxygen, 0.5 mL of methane gas was injected via the T-branch in-line with the brass tube. Thus a 'plug' of methane gas passed through the tube onto the sensor surface, followed by pure air. Sensor resistance was sampled (using the data acquisition system described above) every second for a period of 50 seconds. Resistance data were then transferred via text file to Excel for analysis.

4.3.3 Sensor response versus temperature studies

The temperature-dependent response of a sensor operating in air was investigated by applying a linear voltage ramp to a TGS2611 sensor mounted in the sample flask of the gas manifold system (Chapter 2). The flask was flushed with pure air for a period of 5 minutes, then sealed prior to collecting the resistance-temperature profile. The duration of the voltage ramp was 51.2 seconds, giving the sensor time to equilibrate thermally and thus generate a linear temperature ramp from ambient to ~493 °C (Chapter 3). Sensor resistance was sampled every 0.4 seconds, giving 128 data points over the duration of the scan. Resistance versus temperature data were transferred via text file to Excel for analysis.

4.4 Results and discussion

4.4.1 Sensor response versus oxygen pressure studies

The traces in Figures 4.1 and 4.2 show the response of the TGS2611 sensor to pressures of pure oxygen in the range 0-0.3 atm, at two different temperatures. At a surface temperature of about 100 °C (attained by operating the sensor at ~1.5 V (Chapter 3)), the sensor response (as resistance) over this pressure range is virtually linear with oxygen pressure, yet at ~493 °C (5 V) the response is curved, with a shape that suggests a square root or similar power law-type relationship. This behaviour has been noted previously by many workers [2,3,7,19,20], but few models, if any, have been proposed which explain the behaviour of the sensor at both high and low temperatures. However, earlier work noted in the Section 4.1 makes it clear that the speciation of surface-adsorbed oxygen changes (between 100 and 200 °C) from a diatomic (O_2, O_2^{2-} or O_2^-) to a monatomic (O^- or O^{2-}) species, and so it seems obvious that a model for sensor behaviour has to incorporate a model for both types of oxygen adsorption. Thus a model is proposed here based on the Langmuir adsorption isotherms for dissociative and non-dissociative adsorption, which, with two important approximations can explain the response of a tin oxide gas sensor across its operating temperature range.

Firstly, the relationship between the resistance (R) of the gas sensor and its fractional surface coverage of oxygen (θ_O) must be examined. (θ_O is used to represent the surface coverage of the dominant adsorbed oxygen species, whether monatomic or diatomic, at a given temperature.) Traditionally, most sensor

researchers have concentrated on the relationship between conductance ($G = 1/R$) and oxygen pressure, but resistance can be a more convenient quantity to examine because it increases monotonically with p_{O_2} [2].

The model proposed here seeks to find a simple relationship between R and θ_O, and can be set out as follows.

In general, the flow of charge Q per unit time t through area A of an n-type conductor is given by:

$$Q = -enAv_d t \tag{5}$$

where e is the charge on an electron, v_d is the drift velocity and n is the number of conduction electrons; the negative sign indicates current direction. Current I is defined as $\frac{dQ}{dt}$ and is thus given by:

$$I = -nev_d A \tag{6}$$

For a polycrystalline metal oxide, the macroscopic current will be the sum of the currents through individual crystals. Within a single crystal there are two current paths that should be considered: current flowing perpendicular to the surface through the bulk, or parallel through the surface region. Tin oxide (the most commonly used gas sensor material) behaves as an n-type semiconductor due to the presence of defects [22]. In a typical gas sensor the most common defect is formed by the removal of surface 'in-plane' oxygen anions from the lattice during sensor preparation, which gives rise to a large number of delocalised electrons (i.e. conduction electrons) in the surface region of cross-sectional area A.

The typical ambient air resistance of a TGS2611 gas sensor operating at ~493 °C is ~20 - 30 kΩ. However, when adsorbed oxygen species are removed by evacuation and the subsequent introduction of combustible gas (to remove any remaining oxygen by combustion), the resistance falls to ~10 - 20 Ω, indicating that the surface of the sensor has become metallic in nature, with a high density of surface region conduction electrons. If the sensor temperature is then lowered (e.g. by 200 °C), the resistance of the sensor falls only slightly (by ~5 Ω). If the bulk (perpendicular to surface) current path (due to lattice defects) were significant, the resistance of the sensor would decrease dramatically with increasing temperature, as the previously localised electrons arising from the defects are promoted from the valence to the conduction band. (It is this effect that is used in transition metal oxide-based temperature sensing devices, e.g. thermistors [23].) This evidence strongly suggests that the dominant current path in tin oxide under sub-ambient oxygen pressures is through the surface region. Thus Equation (6) can be rewritten as:

$$I = -n_s e v_d A \tag{7}$$

where n_s is the number of surface region conduction electrons. For a constant applied potential V the resistance of a material is thus:

$$R = \frac{V}{-n_s e v_d A} \tag{8}$$

Since the adsorption of oxygen to the surface of a metal oxide lowers the density of conduction electrons in the surface region, n_s can be replaced with $(n_T - n_r)$, where n_T is the total number of conduction electrons in the metallic, oxygen-free

surface, and n_r is the effective number of such electrons removed by adsorbed oxygen species. Thus the following equation can be written:

$$R = k\left(\frac{1}{n_T - n_r}\right) \tag{9}$$

where k is a collection of the other constants. Since it seems reasonable that n_r must be proportional to θ_O (each adsorbed oxygen atom removes the same amount of electron density from the surface), n_r can be expressed as $bN\theta_O$, where N is the total number of oxygen adsorption sites, and b is the number of electrons withdrawn from the surface per adsorbed oxygen atom (most likely < 1). Now, a function of the form $y = \frac{1}{a - x}$ is remarkably linear when $x << a$ (by about two orders of magnitude or more). Thus if it is assumed that the number of electrons withdrawn from the tin oxide surface by the adsorption of oxygen is significantly smaller than the total number of conduction electrons in the surface, then to an excellent approximation, R can be written in the form:

$$R = Z\theta_O + R_0 \tag{10}$$

where $Z \approx \frac{k}{n_T^2}$ and $R_0 \approx \frac{k}{n_T}$.

This assumption seems reasonable in light of the enormous increase in the resistance of a typical tin oxide gas sensor which occurs when the temperature of an operating sensor is reduced from around 500 °C to ambient temperature. This increase in resistance (from ~10^4 Ω to ~10^8 Ω at 0.2 atm O_2) is due to the readsorption of oxygen, which in turn removes more conduction electrons from

the surface, many more than had originally been removed at the higher temperature. Thus the number of conduction electrons removed under normal operating conditions is indeed orders of magnitude smaller than the total number that it is possible to remove.

The constant R_0 represents the residual resistance of the sensor in its metallic-like state when all surface-adsorbed oxygen has been removed; that is, the bulk resistance of the sensor. As mentioned previously, this is of the order of 10 - 20 Ω, and can often be neglected at near ambient oxygen pressures. The resistance of the sensor can now be expressed as:

$$R(p_{O_2}) = \frac{Z(Kp_{O_2})^x}{1+(Kp_{O_2})^x} + R_0 \qquad (11)$$

If R_0 is neglected, then Z is effectively the resistance of the sensor when $\theta_O = 1$ (maximum monolayer surface coverage) – that is, the maximum resistance R_{max} of the sensor at a given temperature. The response of the sensor can therefore be expressed as:

$$R(p_{O_2}) = \frac{R_{\max}(Kp_{O_2})^x}{1+(Kp_{O_2})^x} \qquad (12)$$

Thus for a tin oxide based sensor with a fixed surface temperature, Equation 12 can be fitted to the sensor response with $x = 1$ for temperatures below about 150 °C and with $x = \frac{1}{2}$ at temperatures above 150 °C.

Figure 4.1(b) is the fit of Equation 12 with $x = 1$ to the low temperature (100 °C) resistance data of Figure 4.1(a). It can be seen firstly that the fit is excellent with

an R^2 coefficient > 0.999, and secondly that the plot is nearly linear over the pressure range plotted. The latter is typical for systems obeying the Langmuir isotherm at low pressures, and can easily be explained by examining Equation 12. When the pressure of oxygen is low, Kp_{O_2} is much smaller than unity and so R is effectively proportional to p_{O_2}. For this approximation to hold, K must also be smaller than 1 for this sensor. Indeed, if the data is plotted using a linearised form of Equation 12 (with x = 1):

$$\frac{p_{O_2}}{R} = \frac{p_{O_2}}{R_{\max}} + \frac{1}{KR_{\max}} \tag{13}$$

then the coefficients K and R_{max} can be extracted for this particular sensor at this temperature. They are found to be 0.3 atm^{-1} and 1.0×10^6 Ω, respectively.

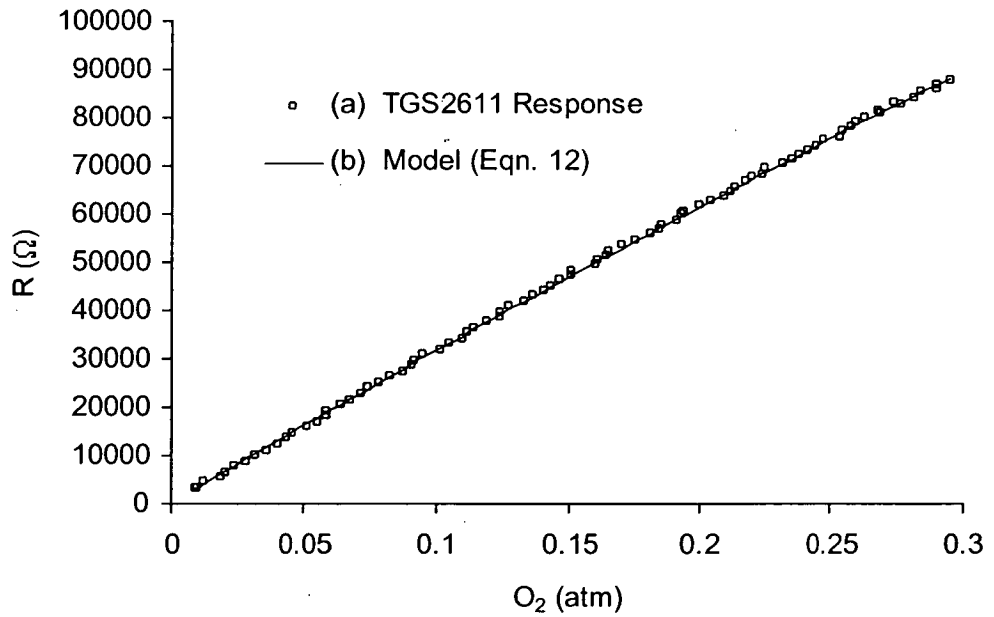


Figure 4.1 TGS2611 oxygen response (0 – 0.3 atm) at surface temperature of ~100 °C.

Figure 4.2(b) is the fit of Equation 12 with $x = \frac{1}{2}$ to the high temperature (493 °C) resistance data of Figure 4.2(a). A good fit is obtained with an R^2 value > 0.99, suggesting strongly that the assumption that dissociation is occurring at this temperature is a valid one. This is also the case when the pressure range is increased to include greater than ambient oxygen partial pressures (0 – 1 atm, Figure 4.3(a)). Figure 4.3(b) shows the close fit ($R^2 > 0.99999$) of Equation 12 to this data, indicating that the model is valid over and beyond the typical operating range of oxygen pressures for a sensor such as the TGS2611. When this high temperature data is plotted using the linearised form of Equation 12 (with $x = \frac{1}{2}$):

$$\frac{p_{O_2}^{1/2}}{R} = \frac{p_{O_2}^{1/2}}{R_{\max}} + \frac{1}{K^{1/2} R_{\max}} \qquad (14)$$

values of K = 0.11 atm^{-1} and R_{max} = 1.0×10^5 Ω are obtained.

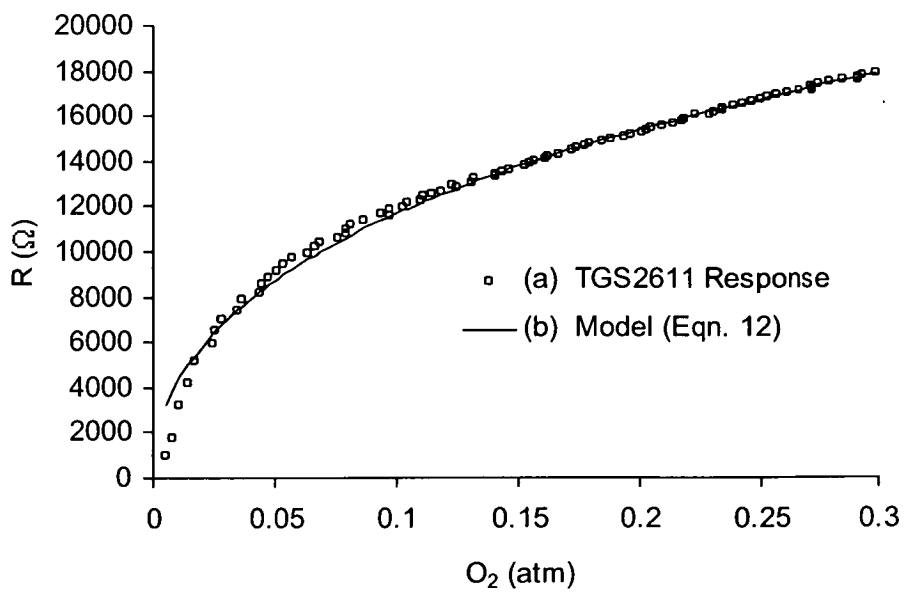


Figure 4.2 TGS2611 oxygen response (0 – 0.3 atm) at surface temperature of ~493 °C.

Many gas sensor researchers have fitted empirical 'power-law' expressions of the form $y = kx^n$ to the oxygen-dependent response of tin oxide gas sensors operating at surface temperatures >700 K [2,3,7,19,20]. The reason that power law expressions can be used to simulate sensor response at sub-ambient pressures of oxygen can be demonstrated by simplifying the dissociative ($x = \frac{1}{2}$) version of Equation 12 as was done earlier for non-dissociative adsorption. When the pressure of oxygen is low, $\left(Kp_{O_2}\right)^{\frac{1}{2}}$ is much smaller than unity and so R is proportional to $p_{O_2}^{\frac{1}{2}}$. This is shown in Figure 4.3(c), where an expression of the form $y = kx^n$ is fitted to the data presented of Figure 4.3(a). It is interesting to note how close the n value is to 0.5, however, although this fit is quite good, it is still inferior to the fit of Equation 12 in its entirety Figure 4.3(b).

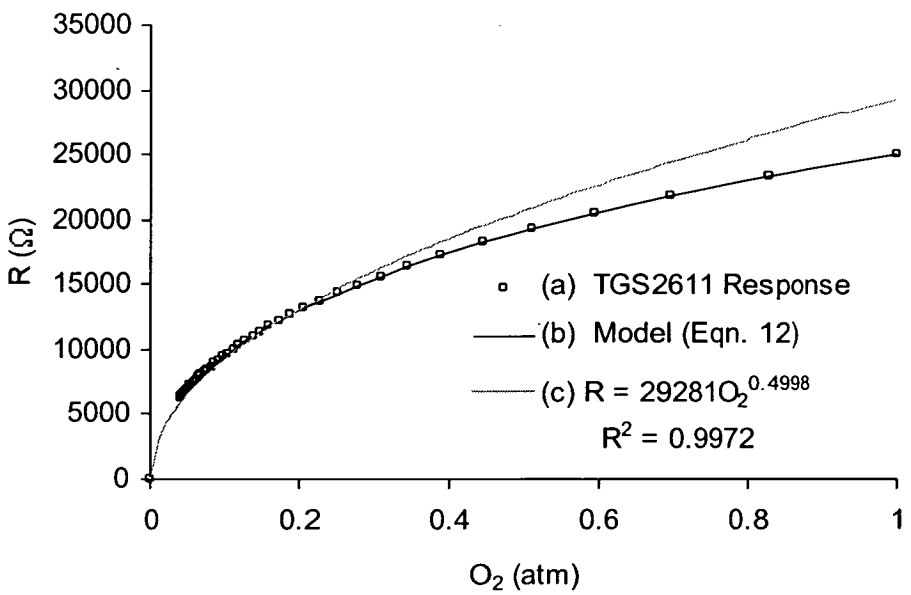


Figure 4.3 TGS2611 oxygen response (0 – 1 atm) at surface temperature of ~ 493 °C.

4.4.2 *Dynamic oxygen adsorption studies*

An alternative method of verifying the model of sensor response to oxygen represented by Equation 12 is to sample sensor resistance with respect to time as oxygen is adsorbed onto a clean sensor surface (all adsorbed oxygen previously removed). This data is shown in Figure 4.4(a), fitted with the time-dependent solution (generated using the fourth order Runge-Kutta technique) of the simultaneous rate Equations 1(a) and 1(b), (Figure 4.4(b)).

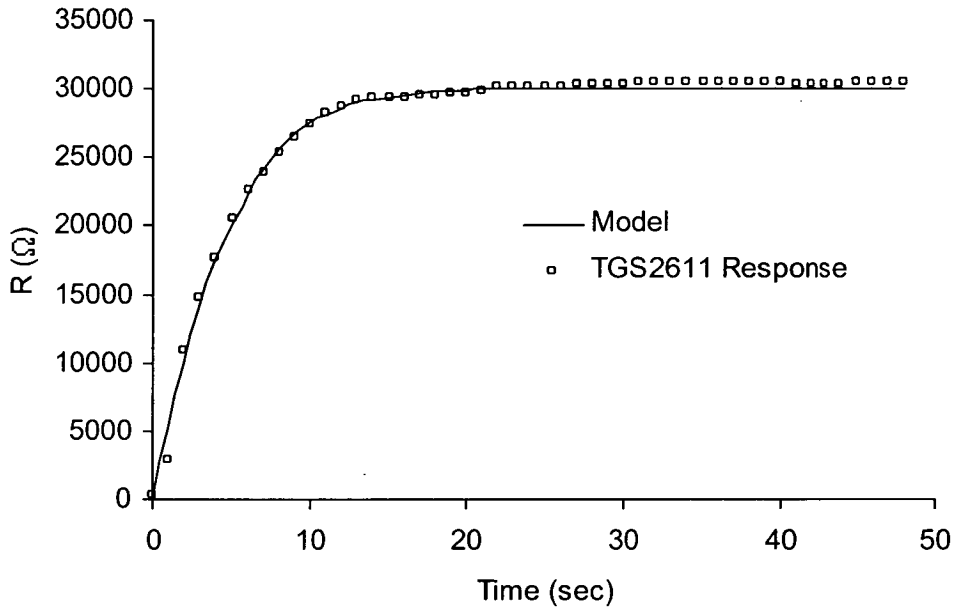


Figure 4.4 TGS2611 dynamic oxygen response (0 – 1 atm) at surface temperature of ~493 °C.

The sensor response flattens out approximately 15 seconds after air is allowed to diffuse back across the sensor surface, indicating that equilibrium has been reached between adsorbed and gas phase oxygen. Minor fluctuations in sensor resistance after equilibrium has been attained are attributable to small variations in

the flow of air across the sensor surface. This further demonstrates the validity of using Langmuir-type adsorption, with the assumption of proportionality of R and θ_O, to model tin oxide gas sensor response to oxygen gas.

4.4.3 Sensor response versus temperature studies

As demonstrated in Figure 3, an expression of the form $y = kx^n$ can be fitted to the TGS2611 oxygen response data with a reasonable fit, particularly over the oxygen pressure range 0 - 0.2 atm, which is where a typical sensor would be operated. This is because for small enough K and p_{O_2}, Equation 12 can be simplified to

$$R(p_{O_2}) = R_{\max}\left(Kp_{O_2}\right)^x \tag{15}$$

using the same rationale as presented earlier. As mentioned in Section 4.2, K can be expressed in the form $K = K_0 e^{\frac{-\Delta H}{RT}}$ and so Equation 15 can be written as:

$$R(T, p_{O_2}) = R_{\max}\left(K_0 e^{\frac{-\Delta H}{RT}} p_{O_2}\right)^x \tag{16}$$

or:

$$R(T) = Ae^{\frac{-x\Delta H}{RT}} \tag{17}$$

where $A = R_{\max}\left(K_0 p_{O_2}\right)^x$. If ΔH is constant over the temperature range of interest, and does not vary significantly with surface coverage, a plot of $\ln R$ vs $1/T$ should be linear with a slope of $\frac{-\Delta H}{R}$ for $x = 1$ and $\frac{-\Delta H}{2R}$ for $x = \frac{1}{2}$.

Such a plot should thus exhibit two distinct linear regions (when $\left(Kp_{O_2}\right)^x$ is small enough to make Equation 15 valid). This is exactly what is observed when data from the sensor response versus temperature experiment is plotted in this way (Figure 4.5). The presence of these two regions indicates two types of oxygen adsorption mechanism occurring over the temperature range plotted (70 - 330 °C). The transition temperature between the two regions is ~160 °C. This is in excellent agreement with the results of EPR, thermal desorption and theoretical studies discussed in the Section 4.1, which suggest a transition temperature between the two mechanisms of between 150 and 200 °C. This evidence reinforces the idea of non-dissociative oxygen adsorption below ~160 °C and dissociative adsorption above this temperature for this particular type of Pd-doped tin oxide sensor.

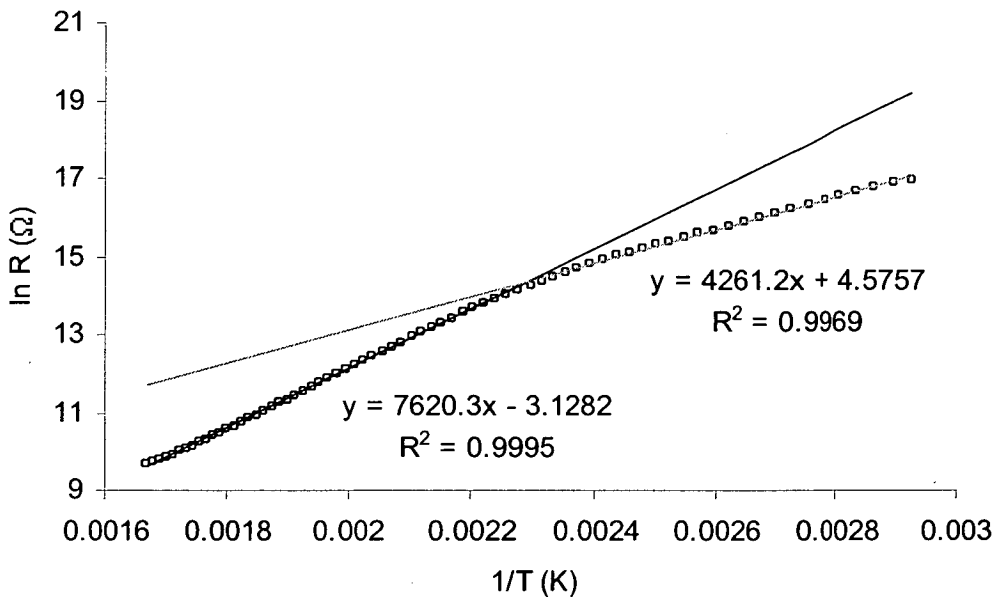


Figure 4.5 lnR vs 1/T plot of TGS2611 ambient air response.

It is important to note that in the low temperature region of this data, the value of Kp_{O_2} can be estimated to be about 0.06 for a typical TGS2611 sensor, using the value of K determined earlier for a different sensor of the same type. This can be considered to be significantly less than one for the purposes of simplifying Equation 12. In the high temperature region, $\left(Kp_{O_2}\right)^{\frac{1}{2}}$ can be estimated to be about 0.14, which means that the use of Equation 15 is not as good an approximation in this case. This can be seen in the lower R^2 value for the fit at higher temperatures.

The enthalpies of oxygen adsorption can be determined from Figure 4.5 to be –35.4 kJ mol^{-1} for low temperature (non-dissociative) adsorption, and –126.7 kJ mol^{-1} for high temperature (dissociative) adsorption. The low temperature value of ΔH is characteristic of physisorption, and is thus in agreement with the idea that there is little covalent interaction with the surface. The high temperature value of ΔH is consistent with chemisorption of the oxygen to the surface.

(A more traditional (and exact) method for obtaining enthalpy of adsorption data for this system would be to find the value of K at several temperatures in each of the two temperature ranges by measuring an isotherm at each temperature, and then finding the slope of a vant Hoff plot. However, for practical reasons this was not attempted – it was found to be too difficult to reproduce the initial sensor response after evacuation at each temperature.)

It is surprisingly difficult to find data in the literature pertaining to the adsorption isotherm(s) of oxygen on tin oxide with which to compare the heats of adsorption found here. In a thermal desorption study which did not fully consider the

dissociation of O_2, Mizusaki et al [24] found three different heats of adsorption for oxygen on sintered tin oxide pellets. These three enthalpies corresponded to three different thermal desorption peaks, at 415, 500 and 735 °C. If it can be assumed that these three peaks correspond to the desorption of O^-, O^{2-} and lattice O, respectively, then the enthalpy calculated for the 415 °C peak (-159 kJ mol^{-1}) can probably be compared with the (dissociative) value of -127 kJ mol^{-1} obtained above, and is in reasonable agreement. This should be qualified by noting that the results presented here are for a commercial, Pd-'doped' SnO_2 sensor and not just sintered SnO_2. However, it seems reasonable from a thermodynamic point of view that if oxygen that adsorbs onto Pd 'spills over' and is ultimately adsorbed onto SnO_2, the enthalpy of adsorption should be the same for doped or undoped tin oxide. Of course, the activation energies for the two processes should be different, and different temperatures are expected for the onset of oxygen dissociation in each case. As noted above, the transition temperature found in this work is very similar to that found by Chang for undoped tin oxide, but it does fall into the lower part of Chang's range [9].

It is somewhat surprising that adsorption onto a SnO_2 sensor can be successfully modelled using Langmuir or pseudo-Langmuir isotherms, because of the non-ideal nature of the polycrystalline sensing material, with its various steps and other surface defects which must depend upon the method of preparation. It may be that one effect of sintering sensor material is to bring about an equivalence between those surface sites which bind oxygen. More fundamental work is needed to determine the adsorption behaviour of oxygen and other gases on tin oxide sensing material– the ability to refer to the adsorption isotherm of O_2 on

SnO_2 or on SnO_2-based sensing material over near ambient O_2 pressures would be extremely useful here, but this information is difficult to find in the literature. The other surprising feature of these results is the sharp delineation of the dissociative and non-dissociative adsorption regions as the sensor temperature is varied. This suggests a relatively small region of overlap between the diatomic and monatomic adsorbed oxygen species. (Note that these results do not provide information about the exact identity of the diatomic and monatomic species.) In this case, techniques such as Raman spectroscopy performed on operating gas sensors might give a more definitive picture of the variation of adsorbed oxygen speciation with temperature.

4.5 Conclusion

The oxygen adsorption behaviour of a commercial Pd / SnO_2 gas sensor has been investigated using a derived relationship between the sensor's surface resistance and the fractional surface coverage of adsorbed oxygen (θ_O). It has been found that oxygen adsorbs non-dissociatively according to a Langmuir-type isotherm at surface temperatures up to ~ 160 °C, and dissociatively (also Langmuir) above that temperature, with enthalpies of adsorption of –35.4 and –126.7 kJ mol^{-1} respectively. The Langmuir-type model can successfully explain sensor response to oxygen at near-ambient pressures over a wide range of temperatures.

References

[1] U. Lampe, M. Fleischer, N. Reitmeier, H. Meixner, J. McMonagle, A. Marsh, New metal oxide sensors: materials and properties, in H. Baltes, W. Göpel, J. Hesse (eds.), *Sensors Update*, Vol. 2 (1996), VCH, Weinheim, pp. 2-36.

[2] P.K. Clifford, Mechanisms of gas detection by metal oxide surfaces, *Ph.D. Dissertation*, Carnegie-Mellon University (1981).

[3] H. Windischmann, P. Mark, A model for the operation of a thin-film SnO_x conductance-modulation carbon monoxide sensor, *J. Electrochem. Soc.* **126** (1979), 627-633.

[4] J.N. Zemel, Theoretical description of gas-film interaction on SnO_x, *Thin Solid Films* **163** (1988), 189-202.

[5] V.M. Aroitiounian, G.S. Aghababian, To the theory of semiconductor gas sensors, *Sens. Actuators B* **50** (1998), 80-84.

[6] B. Fengqi, N. Mengnian, Z. Xianwu, A theoretical study on the gas sensitive characteristics of semiconductive ceramic sensors, *Jpn. J. Appl. Phys.* **35** (1996), 4216-4219.

[7] R.K. Srivastava, P. Lal, R. Dwivedi, S.K. Srivastava, Sensing mechanism in tin oxide-based thick film gas sensors, *Sens. Actuators B* **21** (1994), 213-218.

[8] V.E. Henrich, P.A. Cox, in *The Surface Science of Metal Oxides* (1994), Cambridge University Press, Cambridge, pp. 44-47.

[9] S. Chang, Oxygen chemisorption on tin oxide: correlation between electrical conductivity and EPR measurements, *J. Vac. Sci. Technol.* **17** (1980), 366-369.

[10] N. Yamazoe, J. Fuchigami, M. Kishikawa, T. Seiyama, Interactions of tin oxide surface with oxygen, water and hydrogen, *Surf. Sci.* **86** (1979), 335-344.

[11] P.T. Moseley, D.E. Williams, Oxygen surface species on semiconducting oxides, in: P.T. Moseley, J.O.W. Norris and D.E. Williams (eds.) *Techniques and Mechanisms in Gas Sensing* (1991), Adam Hilger, Bristol, pp. 46-60.

[12] Y. Mizokawa, S. Nakamura, ESR and electric conductance studies of fine powdered tin dioxide, *Jap. J. Appl. Phys.* **14** (1975), 779-788.

[13] T.S. Rantala, V. Lantto, T.T. Rantala, Computational approaches to the chemical sensitivity of semiconducting tin dioxide, *Sens. Actuators B* **47** (1998), 59-64.

[14] Y. Nagasawa, T. Choso, T. Karasuda, S. Shimomura, F. Ouyang, K. Tabata, Y. Yamaguchi, Photoemission study of the interaction of a reduced thin film SnO_2 with oxygen, *Surf. Sci.* **433-435** (1999), 226-229.

[15] Y. Yamaguchi, Y. Nagasawa, S. Shimomura, K. Tabata, Reaction model for methane oxidation on reduced SnO_2 (110) surface, *Int. J. Q. Chem.* **74** (1999), 423-433.

[16] Y. Yamaguchi, Y. Nagasawa, S. Shimomura, K. Tabata, E. Suzuki, A density functional theory study of the interaction of oxygen with a reduced SnO_2 (110) surface, *Chem. Phys. Lett.* **316** (2000), 477-482.

[17] N. Yamazoe, N. Miura, Some basic aspects of semiconductor gas sensors, in S. Yamauchi (ed.), *Chemical Sensor Technology*, Vol. 4 (1992), Kodansha, Tokyo, pp. 19-42.

[18] S.R. Morrison, Selectivity in semiconductor gas sensors, *Sens. Actuators* **12** (1987), 425-440.

[19] Figaro Engineering Inc., The basis of Figaro gas sensor (1998).

[20] H. Ohnishi, H. Sasaki, T. Matsumoto, M. Ippommatsu, Sensing mechanism of SnO_2 thin film gas sensors, *Sens. Actuators B* **13-14** (1993), 677-678.

[21] S. Brunauer, L.E. Copeland, D.L. Kantro, The Langmuir and BET theories, in E.A. Flood (ed.), *The Solid-Gas Interface*, Vol. 1 (1967), Edward Arnold, London, pp.77-103.

[22] V.E. Henrich, P.A. Cox, in *The Surface Science of Metal Oxides* (1994), Cambridge University Press, Cambridge, pp. 149-157.

[23] A.J. Diefenderfer, B.E. Holton, in *Principles of Electronic Instrumentation* (1994), Saunders College Publishing, Philadelphia, pp. 127-131.

[24] J. Mizusaki, H. Koinuma, J. Shimoyama, M. Kawasaki, K. Fueki, High temperature gravimetric study on nonstoichiometry and oxygen adsorption of SnO_2, *J. Solid State Chem.* **88** (1990), 443-450.

Chapter 5

Investigation and simulation of sensor response to *n*-alkanes

5.1 Introduction

Much time and research effort has been invested in attempting to explain the behaviour of tin oxide based gas sensors, particularly the non-linear 'power-law' response to analyte gases. Most of the published work from this research has concentrated on the Schottky barrier layer model of metal oxide conduction [1], oxygen and analyte adsorption theories or combinations of both. Clifford [2] was the first to highlight the limitations of this approach, pointing out that the barrier layer model would predict an exponential relationship between sensor conductance and adsorbed oxygen concentration (something which has never been observed). As the knowledge of adsorbed oxygen species on tin oxide was limited at the time, Clifford went on to suggest that an examination of oxygen adsorption processes and reactions with analyte gases on sensor surfaces could be of use in explaining observations. Although the oxygen speciation problem has been at least partially solved (see Chapter 4), for lack of a better model many researchers have pursued the combined barrier layer model/adsorption theory, publishing many theoretical papers with little or no experimental evidence to verify claims [3-12]. In spite of the lack of a general model of sensor response, it is now widely accepted that the modulation of adsorbed oxygen density by analyte gases is the underlying mechanism responsible for 'power-law' behaviour. However, the details of this mechanism have never been satisfactorily elucidated.

A relatively recent and alternative approach to the investigation of tin oxide sensor behaviour is the simulation of sensor response with computer models that use Monte Carlo (MC) techniques. These techniques simulate physical or imaginary systems using sequences of random numbers [13]. The stochastic approach to

solving complex modelling tasks or mathematical problems is not new, but it is only since the Manhattan project of World War II (where the name 'Monte Carlo', was coined) that they have become fully-fledged numerical methods capable of addressing the most complex applications [13]. Although this technique has seen widespread application in the simulation of chemical processes [14], its application to the study of SnO_2 gas sensor behaviour has been limited [15-20].

Skafidas et al [15] have examined the simulation and modelling of tin oxide-based gas sensor responses to CO and water using MC techniques. In the first of these papers, a model is proposed for the adsorption of oxygen, based on the availability of electrons in the crystal. This model is then extended to consider the contribution of both CO and water vapour. The concentration-dependent responses of sensors to CO and water vapour are not considered, although the time-dependent simulation results appear to be in excellent qualitative agreement with experimental results. A second paper by Skafidas et al [16] examines the modelling and simulation using MC techniques of the more unusual aspects of sensor response, i.e. the undershoot and overshoot of sensor response after exposure to CO. The results indicate that the degree of tin oxide surface reduction appears to affect the oxygen adsorption rate and thus is the mechanism responsible for the observed CO response. The idea has been further investigated by the same group [17] in a study where simulations of CO oxidation on various fractal surfaces were used to explain how the annealing of real tin oxide surfaces could minimise overshoot and undershoot of CO response.

It would appear that the work of Skafidas et al was inspired by an earlier paper by Albano [18], who used Monte Carlo techniques to simulate the oxidation of carbon monoxide on fractal surfaces. The simulation model for the steady state behaviour of the catalytic oxidation of CO on a fractal surface was based on a model (ZGB) originally proposed by Ziff, Gulari, and Barshad [19] for the simulation of a bimolecular reaction on heterogeneous media (Langmuir-Hinshelwood mechanism).

In a different application of Monte Carlo simulations of SnO_2 behaviour, Rantala et al [20] used MC simulations of random barrier networks to study the current-voltage characteristics of polycrystalline semiconductors, in particular SnO_2. This work predicted the non-linearity of current-voltage characteristics of tin oxide based on the geometry of the tin oxide grains in a polycrystalline system. Unfortunately, these results were not compared with any experimental data.

The common conclusion of the various papers reviewed here is that MC techniques provide an alternative method of studying models of adsorption and reaction on surfaces with much greater flexibility than the traditional approach of solving systems of differential rate equations. In contrast, the deterministic approach requires direct information about surface species and sites at the microscopic level, as well as rate constants and initial values of variables which can often not be determined experimentally [21].

In this chapter, work is presented which follows on from Chapter 4, where the oxygen response of an SnO_2-based sensor was shown to arise from the near 'ideal' adsorption behaviour of oxygen altering the number of electrons in the surface conduction path of the sensor material. By taking the model developed for sensor response due to oxygen adsorption and extending it to consider the effect of analyte (combustible) gases, the response of a tin oxide based sensor to a set of 'analytes' has been simulated using MC techniques. The results of the simulation of this 'complete' model and the individual and mixture *n*-alkane ($n = 1 - 7$) responses of a real gas sensor are compared and discussed, providing explanations for the following aspects of sensor behaviour:

- 'power-law' response behaviour
- characteristic R/T profiles of various analytes
- analyte mixture responses

5.2 Theory

Combustible (analyte) gases alter the resistance of SnO_2-based sensors by removing adsorbed oxygen species, thus increasing the density of surface conduction electrons and lowering sensor resistance. There are several possible mechanisms for this process which should be considered.

5.2.1 Competitive adsorption

As explained in Chapter 4, gas phase oxygen may adsorb onto the SnO_2-based sensor surface either non-dissociatively or dissociatively. A second 'analyte' gas introduced into the atmosphere above the sensor surface may either adsorb onto

the surface or remain in the gas phase with no surface interaction. If the analyte gas does adsorb, there are two further possibilities to consider.

The first possibility is that the analyte gas will adsorb onto sites different from those involved with oxygen adsorption, thus having no effect on oxygen surface coverage or the sensor resistance arising from it (from Chapter 4, $R \propto \theta_{O_{surface}}$).

The second possibility is the adsorption of the analyte gas onto the same sites as those involved in oxygen adsorption. In this case, the sensor resistance will alter because the oxygen and analyte will be competing for the same adsorption sites and a net decrease in the surface coverage of oxygen will result. Both of these possibilities can be expressed in terms of adsorption kinetics, as demonstrated below.

If a gas sensor operating at high temperature (eg ~493 °C for a TGS2611) in an atmosphere containing both oxygen and an analyte is considered, then as in Chapter 4, expressions for the dissociative adsorption and desorption of oxygen can be written. At equilibrium, these equations can be equated giving the familiar isotherm equation:

$$k_{a_O} p_{O_2} (1-\theta_O)^2 = k_{d_O} \theta_O^2 \qquad (1)$$

If the non-competitive situation is considered, then a similar set of equations can be written for an analyte gas adsorbing non-dissociatively. In this case the adsorption rate is proportional to the pressure of the analyte and the fraction of unoccupied sites $(1-\theta_A)$, and thus:

$$\frac{d\theta_A}{dt} = k_{a_A} p_A (1-\theta_A) \qquad (2)$$

where k_{a_A} is the rate constant for analyte adsorption. The desorption of analyte is proportional to the fractional surface coverage of analyte θ_A and so:

$$\frac{d\theta_A}{dt} = k_{d_A}\theta_A \tag{3}$$

where k_{d_A} is the rate constant for analyte desorption. As for oxygen, at equilibrium, Equations 2 and 3 can be equated, giving:

$$k_{a_A} p_A (1-\theta_A) = k_{d_A}\theta_A \tag{4}$$

which is the isotherm for analyte adsorption. Equations 1 and 4 represent the non-competitive adsorption situation and show that an analyte has no influence on the surface coverage of oxygen.

If both oxygen and analyte compete for the same adsorption sites, the combined surface coverage of both adsorbates governs the number of empty sites available for adsorption and the isotherm Equations 1 and 4 for oxygen and analyte adsorption at equilibrium become:

$$k_{a_O} p_{O_2} (1-\theta_O-\theta_A)^2 = k_{d_O}\theta_O^2 \text{ (oxygen)} \tag{5}$$

$$k_{a_A} p_A (1-\theta_A-\theta_O) = k_{d_A}\theta_A \text{ (analyte)} \tag{6}$$

Oxygen and analyte surface coverages are now interdependent and so the presence of an analyte in the atmosphere above the sensor surface will decrease the oxygen surface coverage and thus sensor resistance ($R \propto \theta_{O_{surface}}$). This makes the assumption that the analyte gas has no direct effect on sensor resistance when

adsorbed. The validity of this assumption is discussed (with appropriate experimental evidence) in the results and discussion section of this chapter.

5.2.2 Surface reactions – Rideal mechanism

The next mechanism for the removal of adsorbed oxygen to be considered is the potential interaction between an adsorbed oxygen species and an analyte molecule in the gas phase. If an analyte gas molecule collides with an adsorbed oxygen species and reacts, effectively removing the adsorbed oxygen, there will be a decrease in the surface coverage of oxygen, θ_O. If a term for the removal of adsorbed oxygen is introduced into Equation 1, a Rideal-type equation arises:

$$k_{a_O} p_{O_2} (1-\theta_O)^2 = k_{d_O} \theta_O^2 + k_r \theta_O p_A \qquad (7)$$

where k_r is the rate constant for the adsorbed oxygen/gaseous analyte reaction. Increasing analyte gas partial pressure will decrease θ_O and thus sensor resistance.

5.2.3 Surface reactions – Hinshelwood mechanism

The final mechanism for the removal of adsorbed oxygen species removal is the potential reaction of adsorbed oxygen species and adsorbed analyte molecules. As mentioned earlier, analyte gas molecules may adsorb either non-competitively or competitively with oxygen gas molecules onto the sensor surface. In either case, if a reaction occurs between the respective adsorbates then a new term for the removal of both adsorbed oxygen and adsorbed analyte needs to be added to Equations 1 and 4.

In the case of non-competitive adsorption, the equations become:

$$k_{a_O} p_{O_2} (1-\theta_O)^2 = k_{d_O} \theta_O^2 + k_r \theta_O \theta_A \text{ (oxygen)} \quad (8a)$$

$$k_{a_A} p_A (1-\theta_A) = k_{d_A} \theta_A + k_r \theta_O \theta_A \text{ (analyte)} \quad (8b)$$

and for the competitive case:

$$k_{a_O} p_{O_2} (1-\theta_O-\theta_A)^2 = k_{d_O} \theta_O^2 + k_r \theta_O \theta_A \text{ (oxygen)} \quad (9a)$$

$$k_{a_A} p_A (1-\theta_A-\theta_O) = k_{d_A} \theta_A + k_r \theta_O \theta_A \text{ (analyte)} \quad (9b)$$

In both cases, k_r is the rate constant for the adsorbed oxygen/adsorbed analyte reaction. In the non-competitive case (Equation 8), the surface coverage of oxygen is only decreased by the reaction between adsorbed oxygen and adsorbed analyte. However, in the competitive adsorption case, the surface coverage of oxygen is decreased by both the adsorption of analyte and the reaction between adsorbed oxygen and adsorbed analyte.

Sets of equations for multiple analyte gas systems can be written by adding additional terms for the Rideal mechanism or additional equations for the non-competitive and competitive Hinshelwood mechanisms. For example, the set of equations representing the competitive adsorption of oxygen and two analytes, both reacting with oxygen via the Hinshelwood mechanism, would be:

$$k_{a_O} p_{O_2} (1-\theta_O-\theta_A-\theta_B)^2 = k_{d_O} \theta_O^2 + k_{r_A} \theta_O \theta_A + k_{r_B} \theta_O \theta_B \text{ (oxygen)} \quad (10a)$$

$$k_{a_A} p_A (1-\theta_O-\theta_A-\theta_B) = k_{d_A} \theta_A + k_{r_A} \theta_O \theta_A \text{ (analyte A)} \quad (10b)$$

$$k_{a_B} p_B (1-\theta_O-\theta_A-\theta_B) = k_{d_B} \theta_B + k_{r_B} \theta_O \theta_B \text{ (analyte B)} \quad (10c)$$

At this point it should be noted that the sets of equations for the various mechanisms will only apply if the adsorbed species are mobile at all degrees of surface coverage. The validity of this assumption is supported by the successful application of Equation 1 to real oxygen sensor responses in Chapter 4.

The non-linearity introduced by the dissociative adsorption of oxygen occurring with the non-dissociative adsorption of analyte(s) prevents the Hinshelwood equations being solved for θ_O implicitly. This, combined with the many unknown constants involved, makes it impractical to fit the above sets of equations to real sensor responses using regression techniques. To overcome this hurdle and to investigate the applicability of the various mechanisms and their sets of equations, an alternative approach is required.

5.2.4 Monte Carlo simulation techniques

Since the adsorption and reaction mechanisms outlined are stochastic processes, they lend themselves to simulation using the Monte Carlo technique. Simulation of a stochastic process using a MC technique requires the repeated generation of samples of artificial data according to a model, the analysis of each sample and a method of storing results. Most implementations of high level computer languages (eg C, Pascal, etc) include high quality random number generators, and so writing programs that simulate stochastic processes is relatively straightforward, particularly since modern personal computers are capable of processing large amounts of data rapidly.

5.3 Experimental

5.3.1 Investigation of n-alkane effect on sensor resistance in absence of oxygen

The effect of *n*-alkanes on sensor response was investigated using a TGS2611 sensor as described in Chapter 2. Using the gas mixing/dilution system (see Chapter 2), the sample compartment containing the sensor operating at 5 V (~493 °C) was evacuated for a period of five minutes. A small quantity of alkane was then injected to remove any remaining adsorbed oxygen. The sample compartment was then re-evacuated for a period of five minutes to remove any remaining alkane and combustion products. Finally, a quantity of *n*-alkane (to give a pressure of 0.001 atm) was injected and the resistance change of the sensor recorded. This procedure was repeated for several *n*-alkanes.

5.3.2 FTIR monitoring of n-alkane oxidation

Oxidation of ethane by a TGS2611 gas sensor was monitored using the FTIR instrument described in Chapter 2 fitted with a gas cell to which a TGS2611 sensor was connected using a short piece of tubing. With no voltage applied to the gas sensor, a quantity of ethane was injected into the gas cell and a background (single beam) spectrum was collected. A 2 V potential was then applied to the gas sensor heater and three absorbance spectra were collected at 20 minute intervals. The heater voltage was then increased to 5 V and the process repeated.

5.3.3 Collection of sensor response data for n-alkanes (static temperature)

Sensor response data for *n*-alkanes (n = 1 - 7) was collected using a TGS2611 sensor. The gas mixing/dilution, data acquisition and control and associated software systems (see Chapter 2) were used to generate and measure the sensor response to concentrations of the various alkanes from 0 – 2500 ppm in air. The TGS2611 was plugged into the sensor circuit board, which was suspended in the 2.34 L sample compartment. With the applied sensor heater voltage set to 5 V (~493 °C surface temperature) via the 'real time' software (see Chapter 2), the sample compartment was flushed with filtered air of a fixed moisture content until a stable 'background' resistance was obtained. The flask was then sealed and the quantity of either gaseous or liquid alkane required to generate a 500 ppm concentration was admitted via a septum using a syringe. Once a stable resistance reading was obtained, the value was recorded and an identical quantity of alkane was injected and the resistance was allowed to stabilise again. This process was repeated until the concentration of alkane in the flask reached 2500 ppm (this technique minimised any problems associated with sensor drift). The sample compartment was then flushed with air until the same 'background' resistance was obtained, in readiness for the next alkane. Randomly selected samples were replicated to ensure reproducibility.

5.3.4 Collection of sensor response data for n-alkanes (dynamic temperature)

Dynamic or *R/T* response data for *n*-alkanes (n = 1 - 4) was collected using the same TGS2611 sensor and set-up as for the static temperature measurements. For

these experiments the sensor temperature was varied linearly from ambient to ~493 °C, and then back to ambient over a period of 102.4 seconds, giving a 9.63 °C/sec rate of change. The sample compartment was flushed with filtered air until a stable 'background' *R/T* profile was obtained. With the sample compartment sealed, the required quantity (2.34 mL) of gaseous alkane was injected via a septum to generate a 1000 ppm concentration. Once a stable *R/T* profile was obtained, the profile was recorded and the sample compartment flushed until the stable 'background' *R/T* profile returned. The procedure was repeated for the other *n*-alkanes.

5.3.5 Collection of sensor response data for binary n-alkane mixtures (static temperature)

Binary mixture response data for ethane, propane and hexane were collected using the same TGS2611 sensor and set-up as for the static temperature measurements. The sensor was operated at ~493 °C and data were collected using the same procedure as that employed in the static temperature response work. Because binary mixture data were being collected, 36 samples were required to generate the complete matrix over the 500 – 2500 ppm range for both gases (each gas had to be sampled in the absence of the other). The sampling regime was organised so that the sample compartment only had to be flushed 6 times, thus minimising the time required and any problems associated with sensor drift. Two binary systems were examined, ethane and propane and ethane and hexane.

5.3.6 *Oxygen adsorption simulation*

Before MC simulations could be developed for the Hinshelwood equations in Section 5.2.3 a simulation for the dissociative adsorption of oxygen onto a surface was developed as the basis for the more complex simulations to follow. This MC simulation was developed in ANSI Pascal due to its excellent readability and great utility in numerical methods such as MC techniques.

The surface was simulated by a one-dimensional array of integers, with a value of 0 representing a vacant site and a value of 1 an occupied site. The program commences with the user being prompted for integer values for the various parameters i.e. k_{a_O}, k_{d_O} and p_{O_2}. All the array elements (sites) are then assigned a value of 0 (clean surface) and the MC simulation executes a sequence composed of adsorption attempts, desorption attempts and a $\theta_{O_{(sample)}}$ calculation. This sequence is repeated *noMCsteps* times, where *noMCsteps* represents the number of MC steps. Adsorption attempts ($n_{iterations} = p_{O_2}.k_{a_O}$) occur as follows. Two array elements are randomly selected. If both elements have a 0 value then 'adsorption' occurs and both elements are assigned a value of 1. If either or both elements are occupied (element value > 0) then both element values are left unchanged. When the adsorption attempts ($n_{iterations} = p_{O_2}.k_{a_O}$) are completed, $n_{iterations} = k_{d_O}$ desorption attempts occur as follows. Two array elements are randomly selected. If both elements have a value of 1 then 'desorption' occurs and both elements are assigned a value of 0. If either or both elements are unoccupied (element value = 0) then both element values are left unchanged. When $n_{iterations} = k_{d_O}$ attempts are completed, the surface coverage of oxygen

$\theta_{O(Sample)}$ is calculated from the number of array elements with a value of 1 and the result added to $\theta_{O(Sum)}$. When all MC steps are complete, the $\theta_{O(Avg)}$ value is calculated by dividing $\theta_{O(Sum)}$ by *noMCsteps* and the result printed. Figure 5.1 is a flow-chart of this program, and the actual code is presented in Appendix 4.

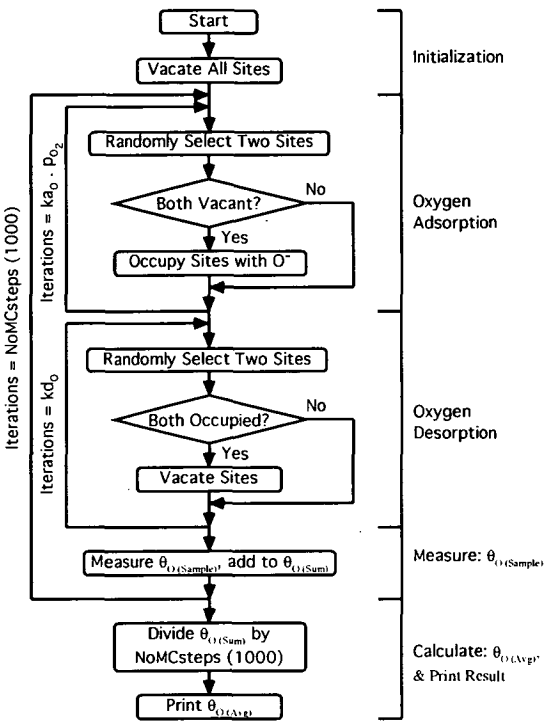


Figure 5.1 Oxygen adsorption/desorption MC simulation program flow-chart

5.3.7 Hinshelwood simulation – single analyte

The program for simulating the competitive adsorption of oxygen and analyte and the surface reaction between those adsorbed species is simply an extension of the oxygen adsorption simulation, with additional steps added for the non-dissociative adsorption of an analyte and the surface reaction mechanism.

Analyte adsorption attempts ($n_{iterations} = p_A .k_{a_A}$) occur as follows. One array element is randomly selected. If the element has a 0 value then 'analyte adsorption' occurs and the element is assigned a value of 2 (this distinguishes analyte adsorption from oxygen adsorption, value = 1). If the element is occupied (element value > 0) then the element value is left unchanged. When $n_{iterations} = p_A .k_{a_A}$ attempts are completed, $n_{iterations} = k_{d_A}$ desorption attempts occur as follows. One array element is randomly selected. If the element has a value of 2 then 'analyte desorption' occurs and the element is assigned a value of 0. If the element is unoccupied (element value = 0) then the element value is left unchanged. When $n_{iterations} = k_{d_A}$ attempts are completed, $n_{iterations} = k_r$ surface reaction attempts occur as follows. Two array elements are randomly selected. If both elements have values > 0 and these two values are not the same (ie one is an oxygen, the other an analyte) then 'reaction' occurs and both elements are assigned a value of 0. If both elements are unoccupied or have the same non-zero value then both element values are left unchanged. As for the oxygen adsorption simulation, the surface coverage of oxygen $\theta_{O(Sample)}$ is calculated at the completion of each MC step and added to $\theta_{O(Sum)}$. Once all MC steps are complete, the $\theta_{O(Avg)}$ is calculated by dividing $\theta_{O(Sum)}$ by *noMCsteps* and the result printed. The flow-chart for this program is shown in Figure 5.2, and the actual code is presented in Appendix 5.

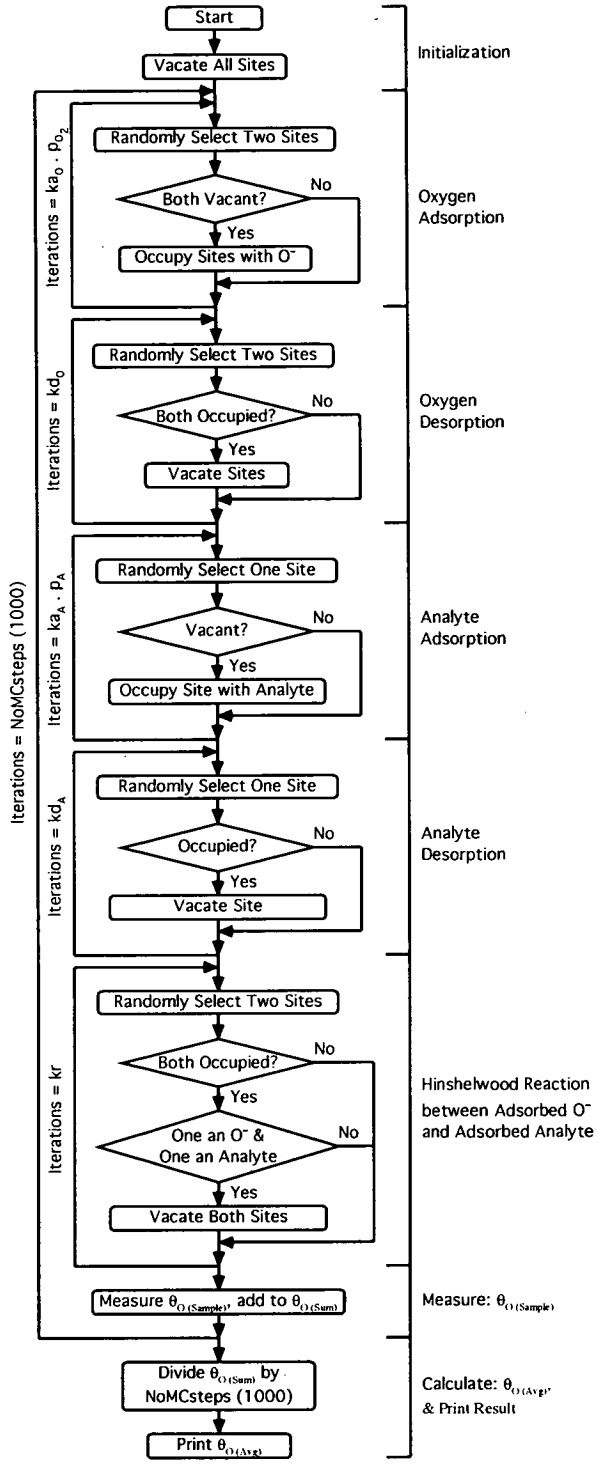


Figure 5.2 Hinshelwood MC program flow-chart

5.3.8 *Hinshelwood simulation – two analytes*

The Hinshelwood simulation program for one analyte was extended to simulate two analytes which adsorb competitively with oxygen and each other, and which both react with adsorbed oxygen. This is accomplished by simply adding additional steps to the single analyte program to account for the adsorption/desorption and reaction of the second analyte. To distinguish the adsorption of the second analyte, an array element was assigned a value of 3 for the successful adsorption of that species. Thus the set of values that a given adsorption site could have were: {0,1,2,3} representing {vacant site, adsorbed oxygen, adsorbed analyte A, adsorbed analyte B}. The code for this program is presented in Appendix 6.

5.4 Results and discussion

5.4.1 *Effect of n-alkane on sensor resistance in absence of oxygen*

The effect of ethane and hexane on the resistance of a TGS2611 sensor operating at 5 V (~493 °C) was investigated after removing adsorbed oxygen by evacuation and combustion (see Chapter 4). The typical 'clean' TGS2611 sensor resistance is ~10 – 20 Ω. If ethane or hexane (0.001 atm) is injected into the evacuated sample compartment containing the sensor, the resistance increases only slightly (~20 Ω). Introduction of additional volumes of ethane or hexane into the flask has no effect on sensor resistance. Therefore any change in sensor resistance observed for *n*-alkane response studies in air is attributed to the removal of surface-adsorbed oxygen.

5.4.2 *n-Alkane oxidation and competitive adsorption*

Oxidation of an alkane (ethane) by adsorbed oxygen on the surface of a TGS2611 sensor was monitored by collecting FTIR spectra of the gas mixture. Oxidation of ethane would result in a decrease in the intensity of the ν(C-H) band of ethane and a corresponding increase in the ν_{as} (antisymmetric stretching) band of CO_2.

Figure 5.3 shows IR spectra of a 1000 ppm ethane/air mixture collected at 20 minute intervals after exposure to a TGS2611 sensor operated at an applied heater voltage of 2 V (~200 °C). After 40 minutes exposure to the sensor, the ratio of the absorbance bands remains unchanged, indicating that no oxidation of ethane has occurred. In contrast, Figure 5.4 shows IR spectra collected in a similar manner, this time with the TGS2611 sensor operating at 5 V (~493 °C). Twenty minutes after exposure there has been a dramatic decrease in the intensity of the ν(C-H) vibration of ethane and a corresponding large increase in the intensity of the ν_{as} vibration of CO_2. Obviously at this temperature, ethane is readily oxidised by adsorbed oxygen on the sensor surface.

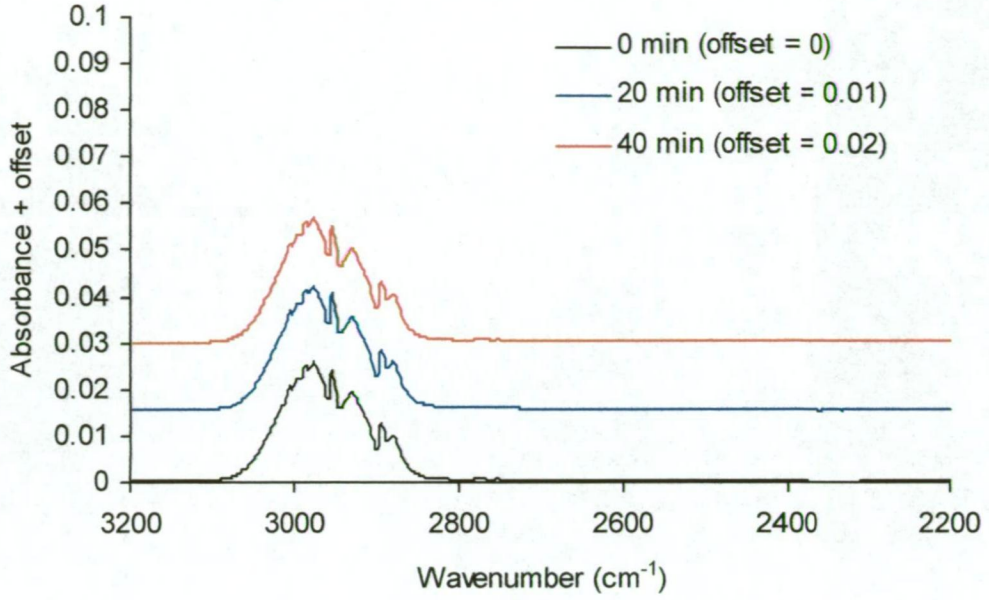


Figure 5.3 FTIR spectra of ethane/air mixture over TGS2611 operated at 2 V (~200 °C)

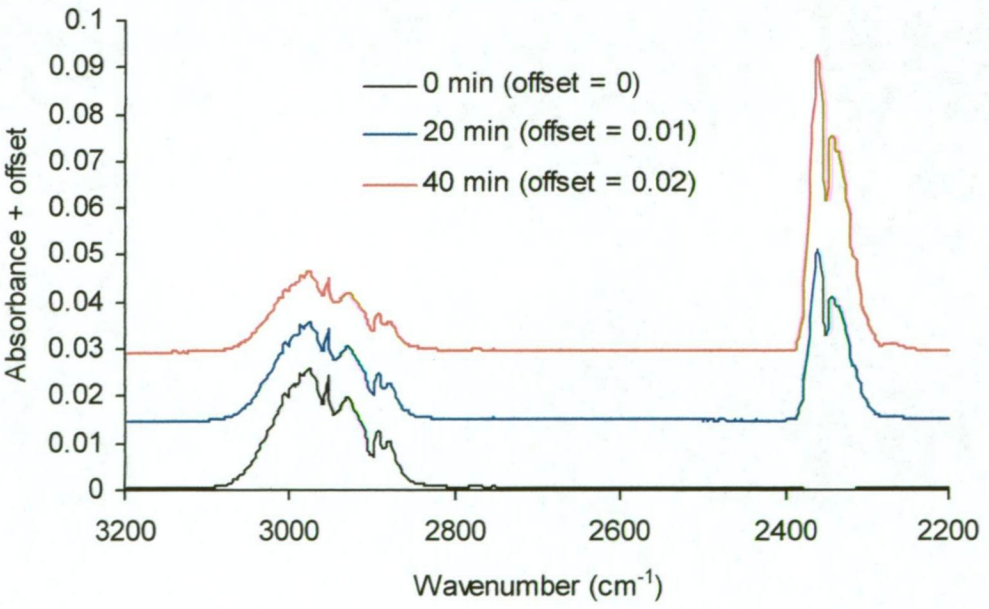


Figure 5.4 FTIR spectra of ethane/air mixture over TGS2611 operated at 5 V (~493 °C)

Resistance/temperature (*R/T*) profiles of a TGS2611 sensor operating in air, a 100 ppm ethane/air mixture and a 100 ppm pentane/air mixture are presented in Figure 5.5. The sensor resistance firstly increases with increasing temperature (attributed to desorption of surface-adsorbed water) and then decreases from ~80 °C onward, due to the thermal desorption of adsorbed oxygen species. Removal of adsorbed oxygen species decreases sensor resistance and so the lower the resistance, the greater the response. The introduction of either ethane or pentane to the atmosphere above the sensor decreased the sensor resistance over the entire operating temperature range.

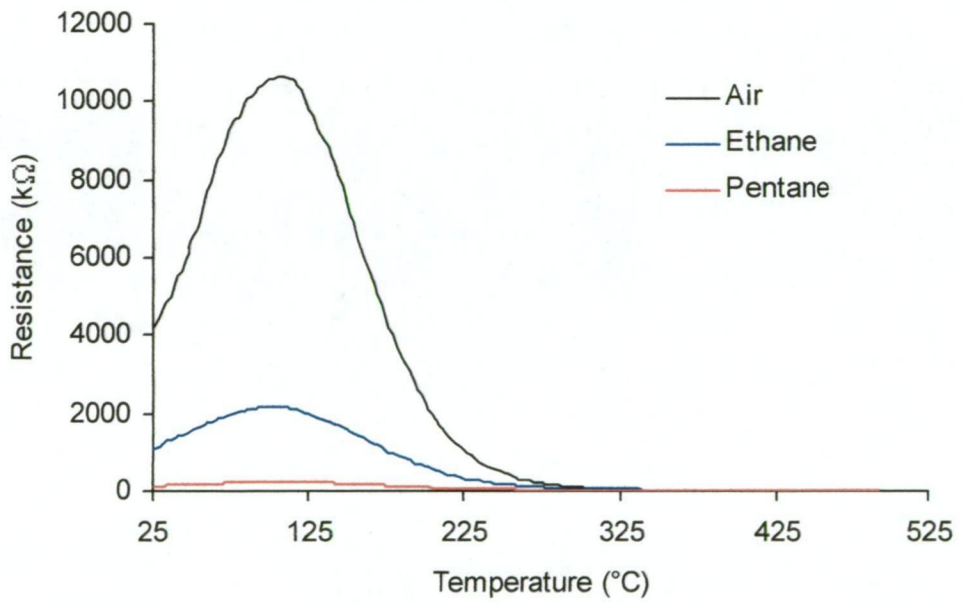


Figure 5.5 R/T profiles of TGS2611 sensor

Obviously the *n*-alkanes have removed adsorbed oxygen species by either a competitive adsorption mechanism or a surface reaction mechanism or both (note: alkanes themselves vary sensor resistance by < 20 Ω in the absence of oxygen). At temperatures below 200 °C, the decease in sensor resistance must be the result

of *n*-alkane molecules adsorbing competitively and thus displacing oxygen species, as the IR work presented found no evidence of ethane oxidation when the sensor was operated at 200 °C even after 40 minutes exposure. The fact that pentane decreases sensor resistance more than ethane at a given temperature (same concentration of both species) also proves that the alkanes are adsorbing onto the sensor surface competitively with oxygen. The fact that pentane had a larger influence on sensor resistance than ethane indicates that the larger molecule displaces more oxygen than ethane.

The IR work showed *n*-alkane oxidation occurring at ~493 °C, thus any decrease in resistance at higher temperatures (> 200 °C) must be the result of both competitive adsorption and surface reaction between adsorbed oxygen species and adsorbed *n*-alkanes. This indicates that the model for sensor response, which includes both competitive adsorption and Hinshelwood surface reactions (expressions 11(a) and 11(b)) is appropriate for simulating a real gas sensor such as the TGS2611.

5.4.3 Oxygen adsorption simulation

Surface coverage vs pressure data produced by the MC oxygen adsorption simulation program is plotted in Figure 5.6. As would occur for a real system exhibiting ideal (Langmuir) adsorption behaviour, the surface coverage (θ) plotted against pressure increases rapidly with increasing pressure and then approaches a limit as the number of surface sites available for adsorption decreases.

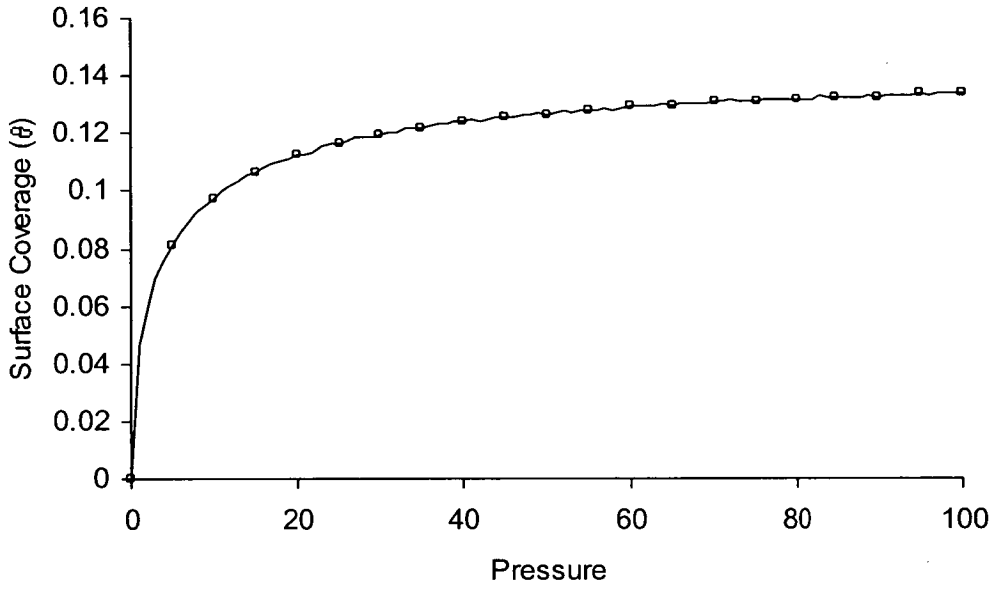


Figure 5.6 Plot of oxygen dissociative adsorption simulation program data

When Equation 1 is linearised, a plot of $\frac{p^{\frac{1}{2}}}{\theta}$ vs $p^{\frac{1}{2}}$ should produce a straight line.

Figure 5.7 is such a plot for the data shown in Figure 5.6. The R^2 value for the fitted line is 0.999 indicating that the program has successfully simulated the ideal dissociative adsorption of a gas onto a surface. By a process of trial and error, values were chosen for the k_{ao} (10) and k_{do} (3000) rate constants to give a value of θ of ~ 0.12. This value is similar to the surface coverage of oxygen (θ_O) on a real TGS2611 sensor (as shown in Chapter 4) and so the rate constant values will be used in the Hinshelwood simulation work that follows. As the program relies on integer arithmetic, the rate constant and pressure values are only representative of real rate constant and pressure values and so cannot be applied to a real gas sensor.

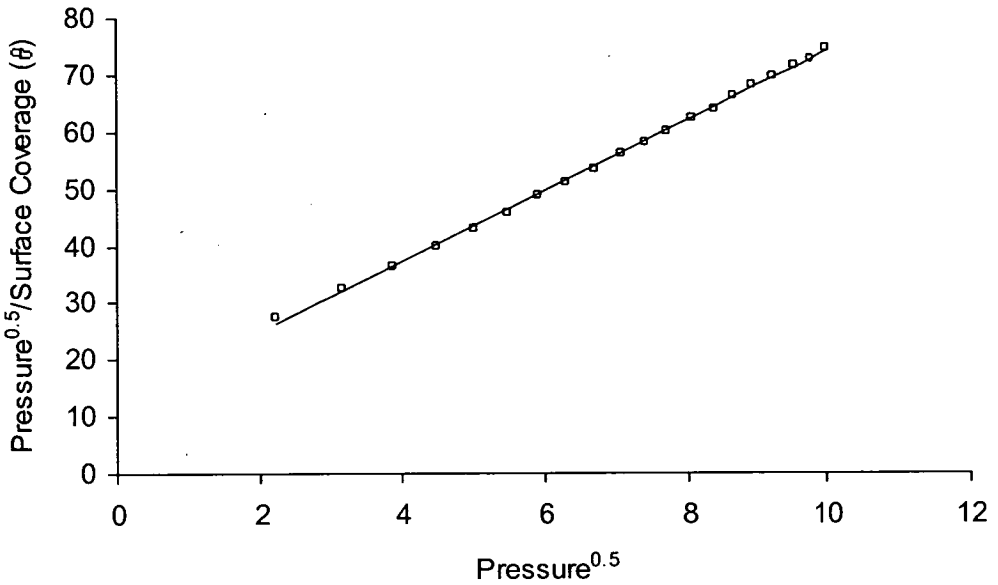


Figure 5.7 Linearised plot of Figure 5.6 data

5.4.4 n-Alkane oxidation and Hinshelwood simulation – single analyte

TGS2611 gas sensor (operating at ~493 °C) response data for various concentrations of *n*-alkanes (500 – 2500 ppm) in air is presented in Figure 5.8. Methane was excluded from these studies because the gas sensor temperature is insufficient at an applied heater voltage of 5 V to produce the maximum oxidation rate which is observed for the longer *n*-alkanes at temperatures below ~493 °C. Trendlines fitted to the *n*-alkane data are 'power-law' functions of the form ($y = kx^n$). The actual equations and their R^2 values are presented in Table 5.1.

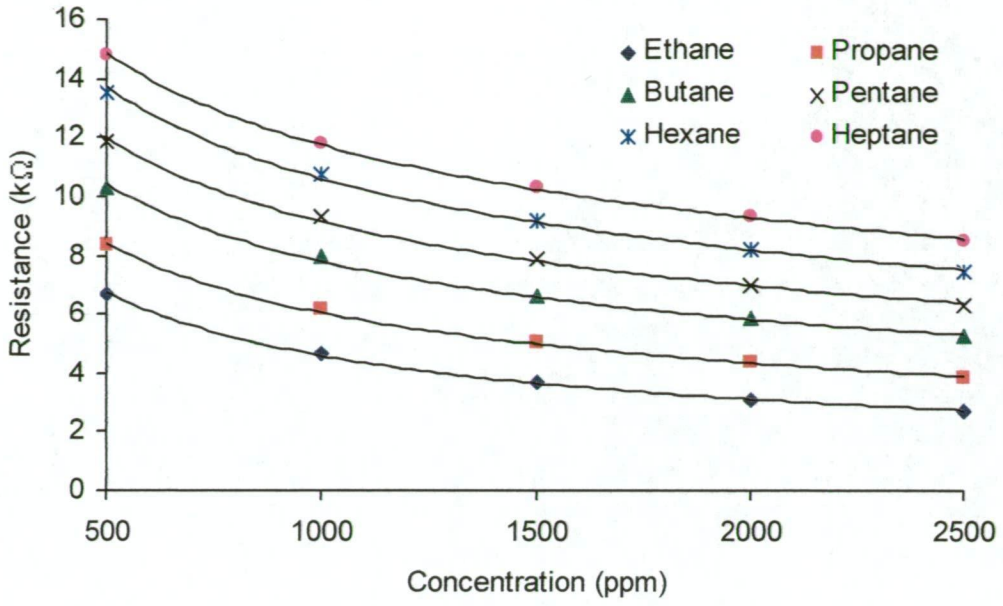


Figure 5.8 TGS2611 (~493 °C) *n*-alkane response data

n-alkane	*equation*	R^2
ethane	$y = 222x^{-0.561}$	0.997
propane	$y = 165x^{-0.478}$	0.998
butane	$y = 142x^{-0.420}$	0.998
pentane	$y = 136x^{-0.391}$	0.997
hexane	$y = 139x^{-0.372}$	0.997
heptane	$y = 124x^{-0.342}$	0.999

Table 5.1 TGS2611 (~493 °C) *n*-alkane response data 'power-law' equations

Clearly, 'power-law' functions fit the response data well; however, close inspection of Figure 5.8 shows that the data points are distributed around the 'power-law' trendlines in a pattern which is common to all six *n*-alkanes. Those points at 500 and 2500 ppm lie below the line, those at 1000 and 1500 ppm lie above the line and finally, those points at 2000 ppm lie on the line. Examination

of the 'power-law' equations in Table 5.1 reveals trends in both the k and n constants. The k and n constant values decrease with increasing chain length, with the largest drop occurring between ethane and propane. The next largest drop is between propane and butane, after which the k values are roughly the same, and the n values decrease more slowly. This suggests that at ~493 °C sensor surface temperature, chain length influences either the adsorption/desorption of the alkane, the rate of oxidation or a combination of these.

The data produced by the MC Hinshelwood simulation program for a series of 'analytes' is plotted in Figure 5.9. The same rate constants determined for the oxygen adsorption simulation were used for this work, while 'analyte/oxygen' pressure ratios were chosen to be similar to those a real gas sensor might be exposed to, remembering that integer values limit the range of ratios that may be used. The k_a adsorption and k_r reaction constants used were the same for all 'analytes', while the k_d desorption constant was increased linearly (ie $k_{d(A)} < k_{d(B)} < k_{d(C)}$ etc). As with the TGS2611 n-alkane response data, 'power-law' trendlines were fitted to the simulation data. The actual equations and their R^2 values are presented in Table 5.2.

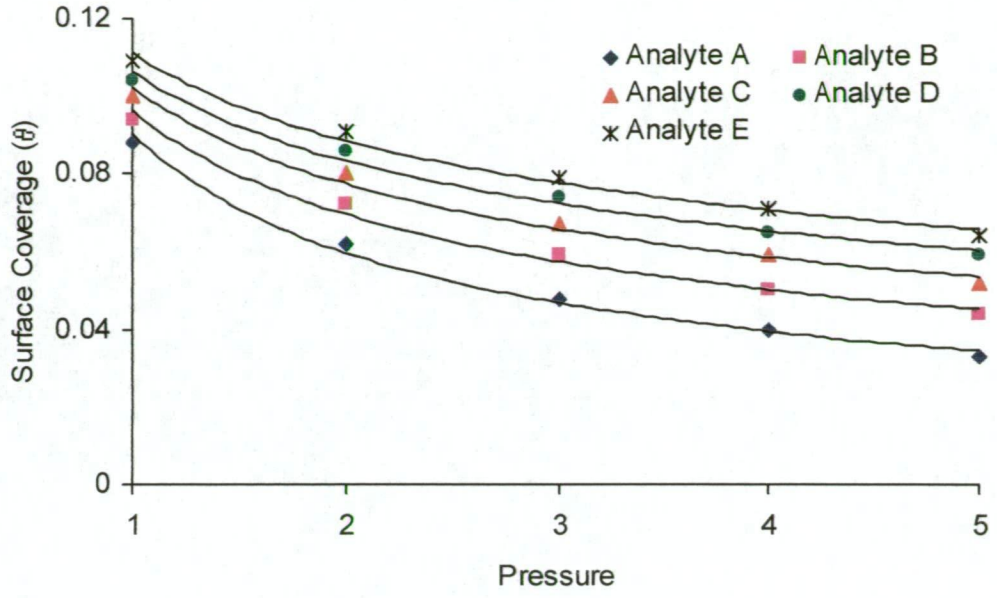


Figure 5.9 Plot of Hinshelwood simulation data for five 'analytes'

analyte	k_{aO}	k_{dO}	k_{aA}	k_{dA}	k_r	*equation*	R^2
A	*10*	*3000*	*100*	*100*	*800*	$y = 0.09x^{-0.6}$	0.99
B	*10*	*3000*	*100*	*160*	*800*	$y = 0.10x^{-0.47}$	0.99
C	*10*	*3000*	*100*	*220*	*800*	$y = 0.10x^{-0.4}$	0.99
D	*10*	*3000*	*100*	*280*	*800*	$y = 0.11x^{-0.35}$	0.99
E	*10*	*3000*	*100*	*340*	*800*	$y = 0.11x^{-0.32}$	0.99

Table 5.2 Hinshelwood simulation data 'power-law' equations

The R^2 values presented in Table 5.2 reveal excellent fits of 'power-laws' to the simulation data. Significantly, the data points lie above and below the trendlines in the same pattern found for the real sensor response data. As will be explained later, the temperature of maximum *n*-alkane oxidation rate decreases with increasing chain length (maximum response ≡ minimum resistance). Thus at

~493 °C, for a given alkane concentration, there would be more of a shorter chain *n*-alkane adsorbed than a longer chain *n*-alkane, and so the shorter alkane produces a bigger response than the longer alkane. This effect was simulated by increasing k_d linearly, thus generating the five 'analyte responses' of Figure 5.9 and Table 5.2. Further examination of Table 5.2 reveals a decrease in the *n* constant of the 'power-laws' fitted to the simulation data. This trend is very similar to that observed for the real gas sensor data 'power-law' fits. The value of *k* remains reasonably constant for the five 'analytes', which is a result of holding k_r constant. This also occurs for the real sensor data when the chain length is *n* 4, suggesting that for these species, the rate of oxidation is very similar and therefore k_r varies little with chain length. Simulations for $n < 4$ would be difficult because both k_d and k_r constants need to be varied simultaneously. For this reason k_d is varied alone, as it is the ratio of k_a/k_d which is important, not the absolute values.

As mentioned earlier, the temperature corresponding to maximum *n*-alkane oxidation rate decreases with increasing chain length. This effect is illustrated in Figure 5.10, which shows a section of the TGS2611 *R/T* profiles for the *n*-alkanes ($n = 2 - 6$) at 1000 ppm. Examination of the ethane, propane and butane profiles shows that the maximum response for ethane occurs at a higher temperature than that for propane, which occurs at a slightly higher temperature than that for butane (this is more obvious on the 'downhill' side of the temperature ramp). As seen for the 'power-law' constants, the temperature of maximum response approaches a constant value (~350 °C) with increasing chain length.

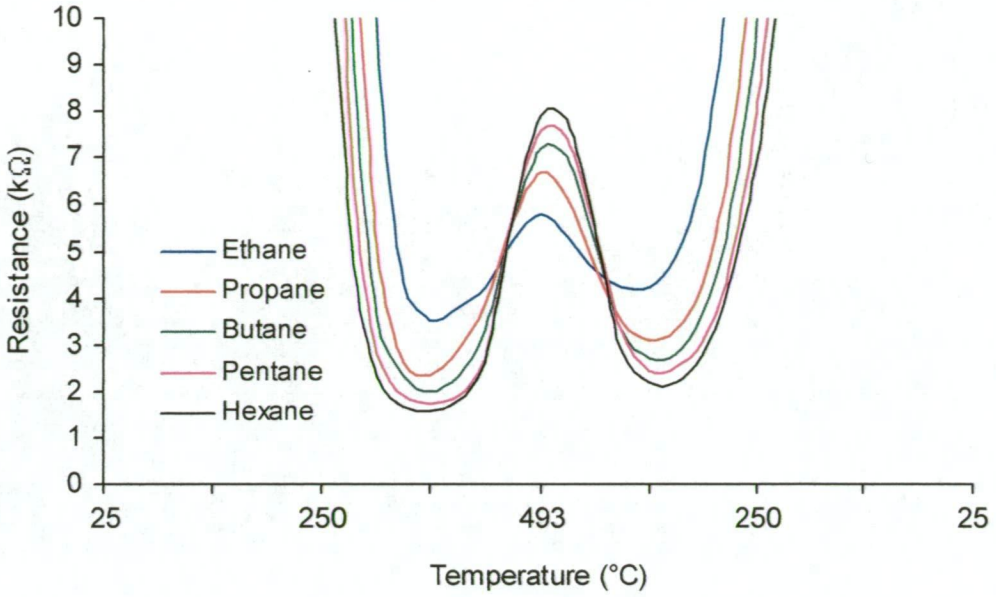
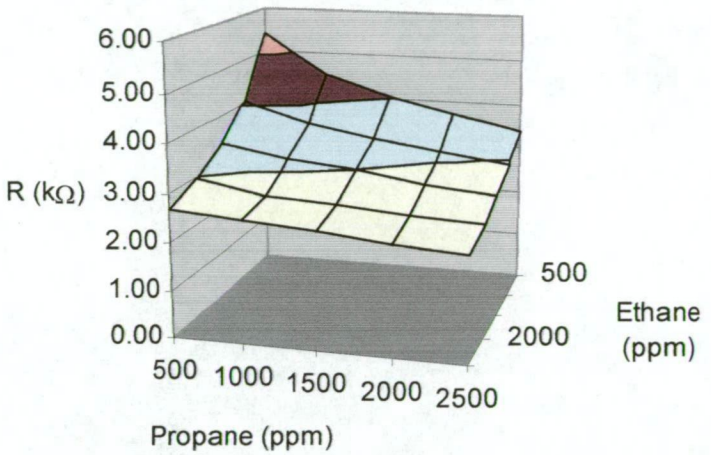


Figure 5.10 TGS2611 *n*-alkane ($n = 2 - 6$, 1000 ppm) *R/T* profiles

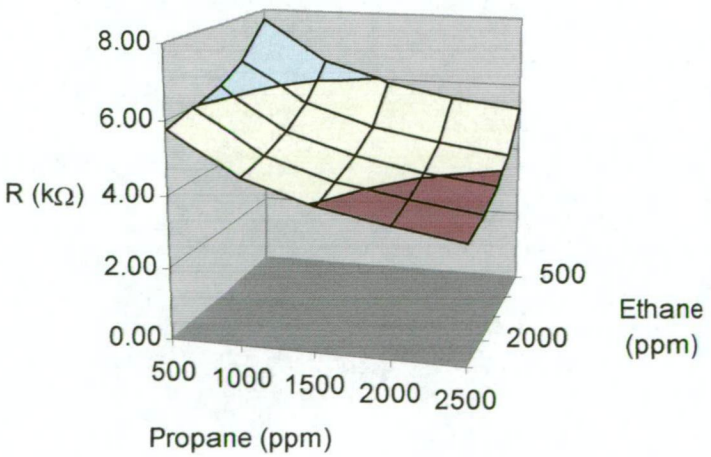
Perhaps the most startling aspect of the *R/T* profiles of Figure 5.10 is the change in the order of response from ~350 – 493 °C. This is the result of the temperature dependence of the k_a/k_d ($K_{analyte}$) ratio varying with chain length, and arises from the variation in the enthalpy of adsorption with chain length (see Chapter 4 for a discussion on enthalpy of adsorption of oxygen species). For example, at ~350°C, more propane is adsorbed than ethane and so its response is greater. However, at ~493 °C, the values of *K* have varied such that now ethane is more strongly adsorbed than propane, etc. Once again, as chain length increases, a limit in response is approached as the adsorption and reaction behaviour of the *n*-alkanes becomes more alike.

5.4.5 *n*-Alkane oxidation and Hinshelwood simulation – binary mixtures

A naïve approach to investigating the response of a sensor to a binary mixture of *n*-alkanes would be to assume that the responses of the individual components combine linearly. As shown in Figure 5.11 this approach fails, with the linear combination of individual responses underestimating the magnitude of response across the entire concentration matrix.



(a)



(b)

Figure 5.11 (a) TGS2611 (5V, ~493 °C) ethane and propane mixture response; (b) Linear combination of TGS2611 (5V, ~493°C) individual ethane and propane responses

As shown in Section 5.4.4, *n*-alkanes remove adsorbed oxygen and thus generate responses by competitively adsorbing and reacting through a Hinshelwood mechanism. It is therefore reasonable to assume, in a mixture of two *n*-alkanes with air, that the alkanes compete with one another for adsorption sites, therefore greatly increasing the complexity of the response mechanism. The effect of alkanes adsorbing competitively can be seen by comparing Figure 5.11 (a) and (b). Obviously the real mixture response is dominated by the more strongly adsorbing analyte, which in this case is ethane. The opposite occurs in the linear combination plot (Figure 5.11 (b)) where propane has dominated the response as expected, since its individual response is greater than that of ethane.

A plot of data produced by the competitive Hinshelwood simulation for two analytes is presented in Figure 5.12 (a), along with a plot of the linear combination of the individual responses (Figure 5.12(b)).

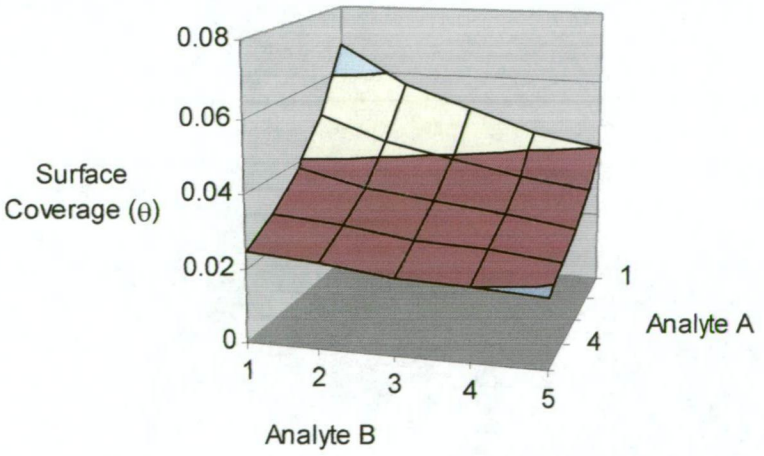


Figure 5.12(a) Competitive Hinshelwood simulation of binary analyte mixture

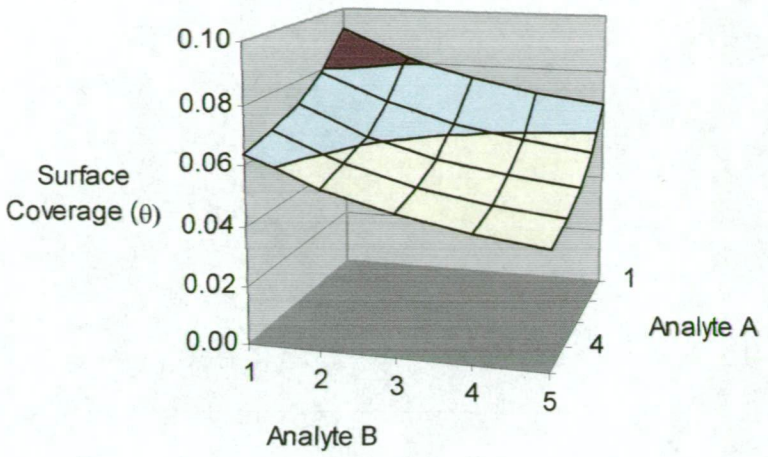


Figure 5.12(b) Linear combination of individual Hinshelwood simulation responses

As occurred for the real sensor mixture response, the linear combination of individual simulation responses underestimated the magnitude of response across the entire concentration matrix. Of more significance is the observation that the simulation mixture response is dominated by the more strongly adsorbing analyte (in this case analyte A), just as occurred for the real sensor mixture response.

These observations suggest that the response mechanism for a binary mixture is indeed a three-way competitive Hinshelwood system, and helps to explain why SnO_2 based sensor responses to mixtures are so complex and have proved so difficult to model even empirically.

5.5 Conclusion

Detailed investigations of sensor response to *n*-alkane/air mixtures using both IR and resistance techniques reveal that alkanes adsorb onto the surface of SnO_2-based sensors but have no effect on the resistance themselves. Therefore the removal of adsorbed oxygen species either by displacement, combustion or both is the underlying mechanism responsible for the non-linear response (resistance) behaviour of these devices. With the knowledge that oxygen exhibits near-ideal adsorption behaviour on gas sensors, a system of equations has been developed to model sensor response by considering the competitive adsorption of oxygen and *n*-alkanes, and the reactions between them on the sensor surface.

Because the surface coverage of oxygen is the only measurable variable, the equation system representing sensor response is under-determined and therefore the fitting of those equations to real responses is impossible. Using the stochastic approach to solving such problems, programs have been developed here that use Monte Carlo techniques to simulate sensor responses to oxygen, analyte/oxygen mixtures and binary analyte/oxygen mixtures.

These simulations produce data analogous to real sensor responses. By carefully examining the real responses and the simulation data, an understanding of the effect of the various equation parameters has been obtained. Now, many of the frequently observed features of static and dynamic gas sensor responses to *n*-alkanes in air can be explained. Although the parameters governing the simulations could not be directly applied to fitting of the equations to real sensor responses, there is strong evidence that the mechanism of sensor response to *n*-

alkanes involves Hinshelwood-like reactions between competitively adsorbed species.

With further research it may be possible to develop a combined simulation/fitting system to directly determine the many parameters (eg k_a, k_d, k_r) for both oxygen and analyte(s)) governing the underlying response mechanisms, using custom sensor designs. For example, although the design of sensors has focussed on miniaturisation in order to minimise heater power consumption, large (> 1 cm^2) sensors may be more useful in determining response mechanisms. By conducting traditional adsorption experiments using physical methods with large areas of sensor material, the adsorption parameters (k_a and k_d) and surface coverages (θ_O and θ_A) of both oxygen and analyte gases could be determined while simultaneously measuring sensor resistance. This would be of great benefit when studying the responses of analytes that not only influence adsorbed oxygen concentration but also directly affect surface conduction electron density and therefore sensor resistance.

References

[1] S.R. Morrison, in *The Chemical Physics of Surfaces* (1977), Plenum, New York, pp. 195-219.

[2] P.K. Clifford, Mechanisms of gas detection by metal oxide surfaces, *Ph.D. Dissertation*, Carnegie-Mellon University (1981).

[3] H. Windischmann, P. Mark, A model for the operation of a thin-film SnO_x conductance-modulation carbon monoxide sensor, *J. Electrochem. Soc.* **126** (1979), 627-633.

[4] J.N. Zemel, Theoretical description of gas-film interaction on SnO_x, *Thin Solid Films* **163** (1988), 189-202.

[5] V.M. Aroitiounian, G.S. Aghababian, To the theory of semiconductor gas sensors, *Sens. Actuators B* **50** (1998), 80-84.

[6] B. Fengqi, N. Mengnian, Z. Xianwu, A theoretical study on the gas sensitive characteristics of semiconductive ceramic sensors, *Jpn. J. Appl. Phys.* **35** (1996), 4216-4219.

[7] N.P. Maksimovich, Relationship between electrical conductivity of metal oxide sensor and catalytic conversion of substances, *Sens. Actuators B* **13-14** (1993), 600-601.

[8] R.K. Srivastava, P. Lal, R. Dwivedi, S.K. Srivastava, Sensing mechanism in tin oxide-based thick film gas sensors, *Sens. Actuators B* **21** (1994), 213-218.

[9] H. Ohnishi, H. Sasaki, T. Matsumoto, M. Ippommatsu, Sensing mechanism of SnO_2 thin film gas sensors, *Sens. Actuators B* **13-14** (1993), 677-678.

[10] A.D. Brailsford, M. Yussouff, E.M. Logothetis, Theory of gas sensors, *Sens. Actuators B* **13-14** (1993), 135-138.

[11] E.Y. Gutman, I.A. Myasnikov, The kinetic semiconductor gas-sensor conduction model and its practical use in gas analysis, *Sens. Actuators B* **13-14** (1993), 687-689.

[12] N. Bârsan, A. Tomescu, The temperature dependence of the response of SnO_2-based gas sensing layers to O_2, CH_4 and CO, *Sens. Actuators B* **26-27** (1995), 45-48.

[13] Computational Science Education Project (CSEP), *Introduction to Monte Carlo Methods*, *Electronic Book,* http://csep1.phy.ornl.gov/Bmap.html (2000).

[14] E.V. Albano, The Monte Carlo simulation method: a powerful tool for the study of reaction processes, *Hetero. Chem. Rev.* **3** (1996) 289-418.

[15] P.D. Skafidas, D.S. Vlachos, J.N. Avaritsiotis, Modelling and simulation of tin oxide based thick-film gas sensors using Monte Carlo techniques, *Sens. Actuators B* **18-19** (1994), 724-728.

[16] P.D. Skafidas, D.S. Vlachos, J.N. Avaritsiotis, Modelling and simulation of abnormal behaviour of thick-film tin oxide gas sensors in CO, *Sens. Actuators B* **21** (1994), 109-121.

[17] D.S. Vlachos, J.N. Avaritsiotis, Simulation of the effect of fractional dimension on tin-oxide-based gas sensors, *Sens. Actuators B* **24-25** (1995), 495-498.

[18] E.V. Albano, Monte-Carlo simulation of the oxidation of carbon monoxide on fractal surfaces, *Surf. Sci.* **235** (1990), 351-359.

[19] R.M. Ziff, E. Gulari, Y. Barshad, Kinetic phase transitions in an irreversible surface-reaction model, *Phys. Rev. Lett.* **56** (1986), 2553-2556.

[20] T.S. Rantala, V. Lantto, T.T. Rantala, Computational approaches to the chemical sensitivity of semiconducting tin dioxide, *Sens. Actuators B* **47** (1998), 59-64.

[21] D. Duca, L. Botár, T. Vidóczy, Monte Carlo simulation of ethylene hydrogenation on Pt catalysts, *J. Catal.* **162** (1996), 260-267.

Chapter 6

Concluding remarks

6.1 Achievements

This thesis has documented research into the surface science and response mechanisms of SnO_2-based gas sensors. In the pursuit of solutions to two fundamental problems that have limited the further application of these sensors in qualitative and quantitative gas analysis, many aspects of sensor response behaviour have been examined. The findings of this work can be summarised as follows:

- The surface temperature of commercial gas sensors can be measured radiometrically to within ± 5 K of the actual temperature.
- The heater voltage-temperature relationships of two types of commercial sensor (Figaro TGS813 and TGS2611) were found to be pseudo-linear, with the temperatures significantly higher than most previously reported estimates.
- Observations of sensor response at various temperatures in the absence of gases have revealed that the dominant conduction path in SnO_2 is through the surface region.
- It was shown that oxygen exhibits nearly ideal (Langmuir) adsorption behaviour on SnO_2-based gas sensors.
- A new model based on adsorption and conduction theories was proposed for the oxygen response of a commercial SnO_2-based sensor (TGS2611) over a wide temperature range.
- Oxygen response studies using the new response model have confirmed earlier work regarding the speciation of adsorbed oxygen.

- Analyte (*n*-alkane) response studies conducted in the absence of other gases have revealed that these species have a negligible effect on sensor resistance compared with oxygen, confirming that the change in sensor resistance observed for *n*-alkane response in air is attributable to the removal of surface-adsorbed oxygen.
- FTIR studies of atmospheres above an operating gas sensor (TGS2611) have revealed that *n*-alkanes do not undergo oxidation until sensor surface temperatures exceed ~200 °C.
- *n*-Alkanes adsorb competitively with oxygen onto the sensor surface.
- The mechanism responsible for the combustible gas response of a commercial sensor (TGS2611) is a competitive adsorption (Hinshelwood) mechanism.
- Monte Carlo-type computer simulations were successfully used to simulate sensor response to single *n*-alkanes and binary mixtures of these gases.
- Detailed information has been acquired about the adsorption and kinetic behaviour of oxygen and *n*-alkanes on SnO_2-based sensors.
- Many aspects of the static and dynamic temperature responses of these devices have finally been explained.

Although many findings have been made and problems solved in this work, there remains much work to complete before metal oxide based gas sensors are applied as qualitative and quantitative gas analysis devices.

6.2 Future work

Further research is required to develop a method of fitting the competitive adsorption (Hinshelwood) model to real sensor responses. This is crucial if sensors are to achieve their full potential in qualitative and quantitative mixture analysis. Probably the best approach to this problem involves the design of new sensors from which information can be obtained about the relative amounts of adsorbed species. For example, the construction of large sensors would permit the simultaneous measurement of sensor resistance and gas adsorption kinetic parameters using traditional surface science techniques.

An alternative approach that may also bear fruit is the simultaneous measurement of sensor resistance and surface temperature. Measurement of the change in sensor surface temperature due to the surface-catalysed reactions may provide additional information about the identity of analyte species.

There are many opportunities for further research into these fascinating sensors and hopefully others will find the information and ideas presented in this thesis of great utility.

Appendices

1. Real time package - Tattletale program

```
//** Tattletale Model 8 program to control sensor temperature,
//** sample sensor resistance and send data packets to MAC program
//** for plotting. Accepts data packets from MAC program to
//** configure parameters.

//** Compiler directives

      model 800          //** Use model 8
      extension KbHit,KbChar
      extension HPSleep      //** Enable extra functions
      CBreak exit            //** If Ctrl-C Pressed => Main Menu

//** Constants and variables

      smin! = 0
      smax! = 5000           //** Sensor heater voltage limits
      samtime = 10           //** Initial sampling interval
      choice = 0             //** Select parameter
      dacbit! = 0.051        //** FP value
      dacval = 0
      senval = 0             //** DAC and ADC variables
      npts2 = 256
      npts = npts2/2
      inc = 0
      pt = 0
      updwn = 1              //** DAC triangle waveform variables

      SDO 255,8              //** Set DAC to Max Volts **
      smplout = 0            //** Packet flow control
      HPSleep(0)             //** Init sleep

//** Main Loop of Program, Detects User Input etc **

loop1:      dacval = INT((smin+(((smax-smin)/npts)*inc))*dacbit)  //** T. Wave
      SDO dacval,8           //** Output wave value to D/A Convertor **
      senval = chan(4)/16    //** Sample log of resistance
      if smplout > 0 print "(",#3D,pt," ",#4D,senval,")",\13;  //** Send Packet
      inc = inc + updwn
      if inc = (npts) updwn = -1
      if inc = 0 updwn = 1
      pt = pt + 1
      if pt = (npts2) pt = 0     //** Generate Triangle Wave **

      iff KbHit() > 0            //** MAC Program control
       choice = KbChar()
       if choice = 81 goto exit  //** Quit
       if choice = 80 gosub prmts  //** New parameters
```

```
        if choice = 71 smplout = 1   //** Send data packets
        if choice = 83 smplout = 0   //** Stop sending data packets
      endif
      HPSleep(samtime)       //** Generate sampling delay

      goto loop1          //** Repeat main loop

//** Get New Parameters from User **

prmts:      input""smin
      input""smax
      input""samtime       //** Get new parameters
      if samtime < 5 samtime = 5
      if samtime > 1000 samtime = 1000
      if smin < 0 smin = 0
      if smin > 5000 smin = 5000
      if smax < 0 smax = 0
      if smax > 5000 smax = 5000 //** Check their within limits
      return               //** Return to main loop

//** Quit instruction so exit program **

exit:  SDO 255,8     //** Set DAC Output to Max **
      print"Hope you had a nice day....."      //** Farewell message...
```

2. Real time package - Macintosh program

```
{-------------------------------------------------------------------------------}
{Full Macintosh application with interface-menus dialogs etc}
{displays sensor resistance in real time with auto-scaling}
{snapshot of resistance plot for comparison with other scans}
{real time capture of resistance data to disk with averaging function}
{user configurable settings for sampling time, sensor heater}
{voltage limits, averaging etc}
{-------------------------------------------------------------------------------}

program RealTimeMac;
   uses
      serial, QDOffscreen;            {enable extra OS packages}
   const
      npoints = 256;              {number of data points}
      halfnpoints = 128;             {half the number of data points}
      mini = 32;
      mid = 129;
      full = 256;              {scales used for plotting}
      inputbufsize = 16384;
      portparam = baud9600 + data8 + noparity + stop10;  {serial port settings}
      MenuBarId = 128;
      AppleMId = 128;
      FileMId = 129;
      EditMId = 130;
      ToolsMId = 131;          {resource ID's for menus}
   type
      mdata = array[1..npoints] of real;   {resistance values}
      optiontype = record
            nwaves: integer;
            sinterval: integer;
            minvolt: integer;
            maxvolt: integer;
         end;                {control parameters}
   var
      error: oserr;                {OS error eg serial comms}
      doneflag: boolean;
      scanflag: boolean;
      snapshot: boolean;             {control flags
      scandialog, prefsdialog, thedialog: dialogptr;
      whichwindow: windowptr;        {pointers to various dialogs etc}
      watch: curshandle;             {watch cursor}
      serialOK: boolean;             {Serial Coms OK}
      whitec, black, red, blue, ltblue, green: RGBColor; {plotting colours}
      sampledata, scandata, snapshotdata: mdata;      {sets of data points}
      axismax, axismid, scale: real;     {auto scaling variables}
      graphbox: rect;              {plot area}
      goutputrefnum: integer;
```

```
    ginputrefnum: integer;
    ginputbufhandle: handle;      {serial comms buffers,refs etc}
    incomingdata: boolean;
    thenumstr: str255;            {incoming packets control and data}
    SmpleNo: longint;             {current sample number}
    sensorohms, samplemax, sampleavg, snapshotavg: real;   {sampling vars}
    result: boolean;
    slowfactor: integer;          {plot rate for slower computers}
    options: optiontype;          {actual scan parameters}
    scanprogress: integer;        {disk dump progress}
    myOffGWorld: GWorldPtr;
    offPixMapHandle: PixMapHandle;
    offGWorldbox: rect;           {variables for offscreen drawing}

{-------------------------------------------------------------------}
{Routine for handling text in dialog windows}
{-------------------------------------------------------------------}

  procedure handleText (thedialog: dialogptr; itemno: integer; text: str255);
    var
      itemtype: integer;
      item: handle;
      box: rect;       {dialog variables}
  begin
    getditem(thedialog, itemno, itemtype, item, box);
    SetIText(item, text);     {change text of dialog item}
  end;

{-------------------------------------------------------------------}
{Log base 10 function-not built into OS}
{-------------------------------------------------------------------}

  function log (x: real): real;
  begin
    log := ln(x) / ln(10);
  end;

{-------------------------------------------------------------------}
{Anti-log base 10 function-not built into OS}
{-------------------------------------------------------------------}

  function antilog (x: real): real;
  begin
    antilog := exp(ln(10) * x);
  end;

{-------------------------------------------------------------------}
{Routine to open and configure comm port for program}
{-------------------------------------------------------------------}
```

```
procedure openserialdriver;
   var
      sershkrec: sershk;
begin
   error := opendriver('.AOut', goutputrefnum);
   if error = noerr then
      error := opendriver('.AIn', ginputrefnum);    {OS Happy!}
   if error = noerr then
      begin
         ginputbufhandle := newhandle(inputbufsize);
         HLock(ginputbufhandle);
         error := sersetbuf(ginputrefnum, ginputbufhandle^,inputbufsize);
         with sershkrec do
            begin
               fXon := 0;
               fCTS := 0;
               errs := 0;
               evts := 0;
               fInX := 0;
               fDTR := 0; {set flow-control etc}
            end;
         error := control(goutputrefnum, 14, @sershkrec);
         error := serreset(goutputrefnum, portparam);
      end;  {initialize comm port and buffers etc}
   if error = 0 then
      serialOK := true;
end;

{------------------------------------------------------------------------}
{Routine to send 1 byte out comm port}
{------------------------------------------------------------------------}

   function sendchar (outchar: char): boolean;
      var
         theptr: ptr;          {the byte}
         myparamblock: paramblockrec;
         myPBPtr: parmBlkPtr;
   begin
      sendchar := false;
      if serialOK then
         begin
            theptr := newptr(1); {byte sent out from serial port}
            theptr^ := signedbyte(outchar);
            with myparamblock do
               begin
                  ioRefNum := goutputrefnum;
                  ioBuffer := theptr;
                  ioReqCount := 1;     {one char}
                  ioCompletion := nil;
                  ioVRefNum := 0;
```

```
                ioPosMode := 0;
            end;
        myPBPtr := @myparamblock;
        error := PBWrite(myPBPtr, false);      {write it}
        if error = noerr then
            sendchar := true;
        disposptr(theptr);     {clear pointer}
    end;
end;

{-------------------------------------------------------------------}
{Routine to receive 1 byte from comm port}
{-------------------------------------------------------------------}

function receivechar (var inchar: char): boolean;
  var
    theptr: ptr;
    myreadcount: longint;
    myparamblock: paramblockrec;
    myPBPtr: parmBlkPtr;
    itemtype: integer;
    item: handle;
    box: rect;
begin
    getditem(scandialog, 15, itemtype, item, box);
    receivechar := false;
    if serialOK then
        begin
            inchar := ' ';
            theptr := newptr(1);  {byte read in from serial port}
            myreadcount := 0;
            error := sergetbuf(ginputrefnum, myreadcount);
            SetIText(item, stringof(myreadcount)); {display no-bytes}
            if myreadcount > 0 then
                begin
                    with myparamblock do
                        begin
                            ioRefNum := ginputrefnum;
                            ioBuffer := theptr;
                            ioReqCount := 1;{one char}
                            ioCompletion := nil;
                            ioVRefNum := 0;
                            ioPosMode := 0;
                        end;
                    myPBPtr := @myparamblock;
                    error := PBRead(myPBPtr, false);{read it}
                    if error = noerr then
                        begin
                        inchar := char(integer(theptr^));
                        receivechar := true;
```

```
                    end;
                end;
            disposptr(theptr);
        end;
    end;

{-------------------------------------------------------------------}
{Routine to close comm port}
{-------------------------------------------------------------------}

    procedure closeserialdriver;
    begin
        error := sersetbuf(ginputrefnum, ginputbufhandle^, 0);
        Hunlock(ginputbufhandle);          {restore the old input buffer}
        error := KillIO(goutputrefnum);
        if error = noerr then
            error := closedriver(ginputrefnum);
        if error = noerr then
            error := closedriver(goutputrefnum);
    end;

{-------------------------------------------------------------------}
{Routine to send new sampling parameters to Tattletale}
{-------------------------------------------------------------------}

    procedure sendparameters;
        var
            i: integer;
            thestring: str255;
    begin
        result := sendchar('P');      {Tell Tattletale to receive new parameters}
        numtostring(options.minvolt, thestring);
        for i := 1 to length(thestring) do
            result := sendchar(thestring[i]);
        result := sendchar(chr(13));   {wave form min voltage}
        numtostring(options.maxvolt, thestring);
        for i := 1 to length(thestring) do
            result := sendchar(thestring[i]);
        result := sendchar(chr(13));   {waveform max voltage}
                numtostring(options.sinterval, thestring);
        for i := 1 to length(thestring) do
            result := sendchar(thestring[i]);
        result := sendchar(chr(13));   {sampling interval}
        if options.sinterval < 1000 then
            slowfactor := 8;
        if options.sinterval < 50 then
            slowfactor := 16;
        if options.sinterval < 20 then
            slowfactor := 32; {based on sampling, slow real time display}
    end;
```

```
{-------------------------------------------------------------------}
{Routine to draw blue progress bar as data streamed to disk}
{-------------------------------------------------------------------}

   procedure drawprogress (mydialog: dialogptr; itemno: integer);
      var
         itemtype: integer;
         item: handle;
         box: rect;
         origport: Grafptr; {dialog bits}
   begin
      getport(origport);
      setport(mydialog);
      if mydialog = scandialog then
         begin
            getditem(mydialog, 6, itemtype, item, box);
            framerect(box);
            Insetrect(box, 1, 1);
            if scanprogress > 0 then
               begin
                  box.right := box.left + round(((scanprogress/ (options.nwaves *
                  npoints)) * (box.right –box.left)));
                  RGBForeColor(blue);
                  fillrect(box, gray);
                  RGBForeColor(black)
               end
            else
               fillrect(box, white);  {actually draw it}
         end;
      setport(origport);
   end;

{-------------------------------------------------------------------}
{This is the big routine, it handles all the real time display etc}
{-------------------------------------------------------------------}

   procedure gasgraph (mydialog: dialogptr; itemno: integer);
      var
         i: integer;
         origPort: cGrafPtr;
         origDev: GDHandle;    {variables for plotting offscreen}
   begin
      samplemax := 1;  {Initialise max sample for graph scaling}
      sampleavg := 0;   {Initialise average variable for sample}
      snapshotavg := 0;{Initialise average variable for snapshot}
      for i := 1 to npoints do
         begin
            if sampledata[i] > samplemax then
               samplemax := sampledata[i];
```

```
            if snapshotdata[i] > samplemax then
               samplemax := snapshotdata[i];
            sampleavg := sampleavg + sampledata[i];
            snapshotavg := snapshotavg + snapshotdata[i];
         end;
      scale := 224 / samplemax;
      axismax := (256 / scale);
      axismid := (128 / scale);        {Calculate the scale for plotting}
      sampleavg := (sampleavg / npoints);   {Calculate average for samples}
      snapshotavg := (snapshotavg / npoints);{Calculate average for s'shot}
      setport(mydialog);            {set port to dialog window}
      GetGWorld(origPort, origDev); {save window's graphics port}
      SetGWorld(myOffGWorld, nil);{make offscreen world the current port}
      EraseRect(offGWorldbox);        {initialize its pixel image}
      RGBForeColor(ltblue);
      for i := 1 to 7 do
         begin
         moveto(offGWorldbox.left, offGWorldbox.bottom - i * mini);
         lineto(offGWorldbox.right - 1, offGWorldbox.bottom - i * mini);
         moveto(offGWorldbox.right - i * mini, offGWorldbox.bottom - 1);
            lineto(offGWorldbox.right - i * mini, offGWorldbox.top);
end;              {create graph grid}
      RGBForeColor(black);
      pensize(3, 3);
      moveto(offGWorldbox.left + SmpleNo - 1, offGWorldbox.bottom –
               round(sampledata[SmpleNo] * scale) - 3);
      line(0, 0);
      pensize(1, 1);
      RGBForeColor(green);        {create temperature triangle}
      moveto(offGWorldbox.left, offGWorldbox.bottom);
      lineto(offGWorldbox.left + 128, offGWorldbox.top);
      lineto(offGWorldbox.left + 256, offGWorldbox.bottom);
      for i := 1 to halfnpoints - 1 do        {plot the sampled gas data}
         begin
         RGBForeColor(blue);
         moveto(offGWorldbox.left + (i * 2) - 1,
                     offGWorldbox.bottom - round(sampledata[i * 2] * scale) -
                     2);
         if (sampledata[i * 2] > 0) and (sampledata[i * 2 + 2] > 0) then
         lineto(offGWorldbox.left + i * 2, offGWorldbox.bottom –
                     round(sampledata[i * 2 + 2] * scale) - 2);
         if snapshot then       {plot the snapshot data}
            begin
            RGBForeColor(red);
            moveto(offGWorldbox.left + (i * 2) - 1,
                           offGWorldbox.bottom - round(snapshotdata[i * 2]
                           * scale) - 2);
            if (snapshotdata[i * 2] > 0) and (snapshotdata[I
                           * 2 + 2] > 0) then
            lineto(offGWorldbox.left + i * 2, offGWorldbox.bottom –
```

```
                        round(snapshotdata[i * 2 + 2] * scale) - 2);
      end;
   end;
textfont(4);
textface([]);
textsize(9);    {set up font etc for drawing averages}
textmode(srccopy);
RGBForeColor(blue);
moveto(offGWorldbox.left + 100, offGWorldbox.top + 12);
drawstring(stringof('Sample Avg: ', sampleavg / 1000 : 10 : 2, ' k '));
if snapshot then
   begin
   RGBForeColor(red);
   moveto(offGWorldbox.left + 100, offGWorldbox.top + 24);
   drawstring(stringof('S"Shot Avg: ', snapshotavg / 1000 :10 : 2, ' k  '));
   end;
textfont(0);
textsize(0);
RGBForeColor(black);
framerect(offGWorldbox);        {create graphbox of graph}
SetGWorld(origPort, origDev);  {make window the current port}
CopyBits(GrafPtr(myOffGWorld)^.portBits, GrafPtr(mydialog)^.portBits,
         offGWorldbox, graphbox, srcCopy, nil);
         {***Use CopyBits to transfer the offscreen image to the
         window***}
RGBForeColor(black); {transferred plots now do axis and labels}
for i := 1 to 7 do
   begin
      moveto(graphbox.left + i * mini + 1, graphbox.bottom);
      lineto(graphbox.left + i * mini + 1, graphbox.bottom + 2);
            {bottom side tick marks}
      moveto(graphbox.left, graphbox.top + i * mini + 1);
      lineto(graphbox.left - 3, graphbox.top + i * mini + 1);    {left side tick
         marks}
      moveto(graphbox.right, graphbox.top + i * mini + 1);
      lineto(graphbox.right + 2, graphbox.top + i * mini + 1); {right side
         marks}
   end;
moveto(graphbox.left, graphbox.bottom);
lineto(graphbox.left, graphbox.bottom + 2);
moveto(graphbox.left + full, graphbox.bottom);
lineto(graphbox.left + full, graphbox.bottom + 2);
moveto(graphbox.left, graphbox.top);
lineto(graphbox.left - 3, graphbox.top);
moveto(graphbox.left, graphbox.top + full);
lineto(graphbox.left - 3, graphbox.top + full);
moveto(graphbox.right, graphbox.top);
lineto(graphbox.right + 2, graphbox.top);
moveto(graphbox.right, graphbox.top + full);
```

```
        lineto(graphbox.right + 2, graphbox.top + full);    {create tick marks at
                                                              corners}
        textfont(4);
        textface([]);
        textsize(9);
        textmode(srccopy);
        moveto(graphbox.left - (stringwidth(stringof(round(axismax / 1000) : 6))
                + 4), graphbox.top + 4);
        drawstring(stringof(round(axismax / 1000) : 6));
        moveto(graphbox.left - (stringwidth(stringof(round(axismid / 1000) : 6))
                +4), graphbox.top + mid + 4);
        drawstring(stringof(round(axismid / 1000) : 6));
        moveto(graphbox.left - 10, graphbox.top + full + 4);
        drawstring('0');
        moveto(graphbox.right + 5, graphbox.top + 4);
        drawstring(stringof('Max'));
        moveto(graphbox.right + 5, graphbox.top + mid + 4);
        drawstring(stringof('Mid' : 4));
        moveto(graphbox.right + 5, graphbox.top + full + 4);
        drawstring(stringof('Ambient'));
        moveto(graphbox.left + 1 - (stringwidth('0') div 2),
        graphbox.bottom + 13);
        drawstring('0');
        moveto(graphbox.left + 1 + mid - (stringwidth('128') div 2),
           graphbox.bottom + 13);
        drawstring('128');
        moveto(graphbox.left + 1 + full - (stringwidth('256') div 2),
        graphbox.bottom + 13);
        drawstring('256');              {***create axis numbers & labels***}
        textfont(0);
        textsize(0);     {reset system font etc}
    end;

{-------------------------------------------------------------------}
{Routine, returns item number in dialog}
{-------------------------------------------------------------------}

    function myfiledlg (item: integer; thedialog: dialogptr): integer;
    begin
        myfiledlg := item;
    end;

{-------------------------------------------------------------------}
{Routine to write data to disk, once x-waves are complete }
{-------------------------------------------------------------------}

    procedure scancomplete (thedialog: dialogptr);
        var
            itemtype: integer;
            item: handle;
```

```
        box: rect;
        where: point;
        prompt: str255;
        typelist: SFTypeList;
        reply: SFReply;
        refnum: integer;
        count: longint;
        mystring: Str255;
        i: integer;                    {dialog bits and pieces}
begin
    result := sendchar('S');                    {tell sensor to stop sending data
                                                  packets}
    for i := 1 to npoints do
        scandata[i] := scandata[i] / options.nwaves;      {average the data}
    where.v := 200;
    where.h := 200;                    {position getfile dialog box}
    prompt := 'Save Scan as:';
    SFPPutFile(where, prompt, '', @myfiledlg, reply, 128, nil);        {get file
    info for file save, use own dialog}
    if reply.good then
        begin
            error := create(reply.fname, reply.vrefnum,'APPL', 'TEXT');
            if error = noerr then
                begin
                    error := FSOpen(reply.fname, reply.vrefnum, refnum);
                    if error = noerr then
                        begin
                            error := SetEOF(refnum, 0);  {make sure that the
                                                           file is empty}
                            mystring := stringof('TGS2611', chr(13));
                            count := length(mystring);
                            error := FSWrite(refnum, count, @mystring[1]);{write it}
                            for i := 1 to npoints do
                                begin
                                    mystring := stringof(scandata[i] / 1000 :
                                    10 : 2, chr(13));
                                    count := length(mystring);
                                    error := FSWrite(refnum, count, @mystring[1]);
                                end;                        {write out the data}
                            error := FSClose(refnum);
                            error := FlushVol(nil, reply.vrefnum);
                        end;
                end;
        end;
    scanflag := false;
    scanprogress := 0;
    result := sendchar('G');                    {tell sensor to send data packets}
    drawdialog(scandialog);
    getditem(scandialog, 2, itemtype, item, box);
    hilitecontrol(controlhandle(item), 0);
```

```
    getditem(scandialog, 3, itemtype, item, box);
    hilitecontrol(controlhandle(item), 255);                {reset everything}
  end;
{-------------------------------------------------------------------------}
{Routine which handles and processes data packets from }
{Tattletale}
{-------------------------------------------------------------------------}

  function myfilter (thedialog: dialogptr; var theevent: eventrecord;
                     var itemhit: integer):
    boolean;
    var
      inchar: char;
      temp: longint;
      theptr: ptr;
      myreadcount: longint;
      myparamblock: paramblockrec;
      myPBPtr: parmBlkPtr;          {variables for reading bytes
                                       directly-saves time}
  begin
    if serialOK then
      begin
        inchar := ' ';
        myreadcount := 0;
        error := sergetbuf(ginputrefnum, myreadcount);
        while myreadcount > 0 do
          begin
            theptr := newptr(1);              {byte read in from serial port}
            with myparamblock do
              begin
                ioRefNum := ginputrefnum;
                ioBuffer := theptr;
                ioReqCount := 1;          {one char}
                ioCompletion := nil;
                ioVRefNum := 0;
                ioPosMode := 0;
              end;
            myPBPtr := @myparamblock;
            error := PBRead(myPBPtr, false);        {read it}
            if error = noerr then
              begin
                inchar := char(integer(theptr^));
                if (inchar = '(') or (inchar = ')') then
                  begin
                    if inchar = '(' then
                      begin
                        incomingdata := true;
                        thenumstr := '';    {make the string empty}
                      end;
```

```
                    if inchar = ')' then
                        begin
                            incomingdata := false;
                            if length(thenumstr) = 8 then     {is the length
                                                    right for valid packet?}
                                begin
                                    StringtoNum(copy(thenumstr, 1, 3),
                                    SmpleNo);      {convert the sample
                                    number}
                                    SmpleNo := SmpleNo + 1;
                                    StringtoNum(copy(thenumstr, 5, 4),
                                    temp);
                                    sensorohms := temp;
                                    sensorohms := temp / 1000;
                                    sampledata[SmpleNo] := (10000 /
                                    (antilog((-sensorohms - 0.003) / 1))) -
                                    10000;
                                    handleText(scandialog, 15,
                                    stringof(SmpleNo : 4));
                                    handleText(scandialog, 18,
                                    stringof(myreadcount : 4));
                                    if scanflag then
                                        begin
                                            scandata[SmpleNo] :=
                                            scandata[SmpleNo]
                                            +sampledata[SmpleNo];
                                            scanprogress := scanprogress + 1;
                                            drawprogress(thedialog, 6);
                                            if scanprogress = (options.nwaves
                                                * npoints) then
                                            scancomplete(thedialog);
                                        end;
                                    if SmpleNo mod slowfactor = 0 then
                                        gasgraph(thedialog, 5);
                                        {redraw the plot}
                                end;
                            end;
                        end
                    else if incomingdata then          {data must be a number}
                        thenumstr := concat(thenumstr, inchar);
                end;
            disposptr(theptr);
            error := sergetbuf(ginputrefnum, myreadcount);   {see if more
                                                    characters to read
                                                    in!}
        end;
    end;
  myfilter := false;          {hasn't modified eventrecord so return false}
end;
```

```
{-------------------------------------------------------------------}
{Main scan gas dialog routine to handle events in dialog}
{-------------------------------------------------------------------}

  procedure DoScan;
    var
      itemtype: integer;
      item: handle;
      box: rect;
      i: integer;
      itemhit: integer;
      good: Boolean;
  begin
    slowfactor := 32;
    scanprogress := 0;
    SmpleNo := 1;
    scanflag := false;
    incomingdata := false;
    serialOK := false;              {initialise everything}
    scandialog := getnewdialog(129, nil, pointer(-1));     {Get Scan Gas Dialog}
    getditem(scandialog, 5, itemtype, item, box);
    graphbox := box;
    setditem(scandialog, 5, itemtype, handle(@gasgraph), box);
    getditem(scandialog, 6, itemtype, item, box);
    setditem(scandialog, 6, itemtype, handle(@drawprogress), box);
    {pass the procs for user items}
    getditem(scandialog, 3, itemtype, item, box);
    hilitecontrol(controlhandle(item), 255);                {disable buttons}
    error := NewGWorld(myOffGWorld, 0, graphbox, nil, nil, []);
                        {create offscreen graphics world, }
    if error = noerr then
      begin
        offGWorldbox := myOffGWorld^.portRect;
        offPixMapHandle := GetGWorldPixMap(myOffGWorld);
          {get handle to }
        good := LockPixels(offPixMapHandle);
          { offscreen pixel image and lock it}
        openserialdriver;               {prepare serial port for data transfer}
        if serialOK then                                {no worries so do dialog}
          begin
            showwindow(scandialog);                     {show the dialog}
            result := sendchar('S');  {tell sensor to stop sending data packets}
            sendparameters;                     {update sensors parameters}
            result := sendchar('G');        {tell sensor to send data packets}
            repeat
              ModalDialog(@myfilter, itemhit);
              case itemhit of           {which button etc has been hit}
                2:
                  begin
                    scanflag := true;
```

```
                        scanprogress := 0;
                        for i := 1 to npoints do
                           scandata[i] := 0;
                        getditem(scandialog, 2, itemtype, item, box);
                        hilitecontrol(controlhandle(item), 255);
                        getditem(scandialog, 3, itemtype, item, box);
                        hilitecontrol(controlhandle(item), 0);
                     end;                {start scan button}
                  3:
                     begin
                        scancomplete(scandialog);
                        scanflag := false;
                        scanprogress := 0;
                        drawprogress(scandialog, 6);
                        getditem(scandialog, 2, itemtype, item, box);
                        hilitecontrol(controlhandle(item), 0);
                        getditem(scandialog, 3, itemtype, item, box);
                        hilitecontrol(controlhandle(item), 255);
                     end;                {stop scan button}
                  11:
                     begin
                        snapshot := true;
                        snapshotdata := sampledata;
                     end;                {take snapshot button}
                  otherwise
               end;
            until itemhit = 1;          {done button}
            scanflag := false;
            result := sendchar('S');  {tell sensor to stop sending data packets}
            closeserialdriver;
         end;
      UnlockPixels(offPixMapHandle);      {unlock the pixel image}
      DisposeGWorld(myOffGWorld);         {dispose of offscreen world}
   end;
  disposDialog(scandialog);                              {get rid of dialog}
end;

{-------------------------------------------------------------------------}
{Preferences (parameters) dialog routine to handle events in dialog}
{-------------------------------------------------------------------------}

procedure DoPrefs;
   var
      itemtype: integer;
      item: handle;
      box: rect;
      itemhit: integer;
      thestring: str255;
      thenum: longint;                    {dialog bits}
begin
```

```
prefsdialog := getnewdialog(130, nil, pointer(-1));
            {Get Parameters Dialog}
showwindow(prefsdialog);                          {show the dialog}
SelIText(prefsdialog, 3, 0, 32767);               {select text in first box}
numtostring(options.nwaves, thestring);
getditem(prefsdialog, 3, itemtype, item, box);
SetIText(item, thestring);
numtostring(options.sinterval, thestring);
getditem(prefsdialog, 4, itemtype, item, box);
SetIText(item, thestring);
numtostring(options.minvolt, thestring);
getditem(prefsdialog, 5, itemtype, item, box);
SetIText(item, thestring);
numtostring(options.maxvolt, thestring);
getditem(prefsdialog, 6, itemtype, item, box);
SetIText(item, thestring);                        {show previous parameters}
repeat
   ModalDialog(nil, itemhit);
until (itemhit = 1) or (itemhit = 2);             {wait for event}
if itemhit = 1 then                               {OK button hit}
   begin
      getditem(prefsdialog, 3, itemtype, item, box);
      GetIText(item, thestring);
      stringtonum(thestring, thenum);
      if thenum < 1 then
         thenum := 1;
      if thenum > 10 then
         thenum := 10;
      options.nwaves := thenum;
      getditem(prefsdialog, 4, itemtype, item, box);
      GetIText(item, thestring);
      stringtonum(thestring, thenum);
      if thenum < 5 then
         thenum := 5;
      if thenum > 1000 then
         thenum := 1000;
      options.sinterval := thenum;
      getditem(prefsdialog, 5, itemtype, item, box);
      GetIText(item, thestring);
      stringtonum(thestring, thenum);
      if thenum < 0 then
         thenum := 0;
      if thenum > 5000 then
         thenum := 5000;
      options.minvolt := thenum;
      getditem(prefsdialog, 6, itemtype, item, box);
      GetIText(item, thestring);
      stringtonum(thestring, thenum);
      if thenum < 0 then
         thenum := 0;
```

```
            if thenum > 5000 then
               thenum := 5000;
            options.maxvolt := thenum;
               {get and store each parameter from dialog}
         end;
      disposDialog(prefsdialog);                    {get rid of dialog}
   end;

{-------------------------------------------------------------------------------}
{Routine to init everything upon boot into application}
{-------------------------------------------------------------------------------}

   procedure initthings;
      var   i: integer;
   begin
      doneflag := false;
      watch := getcursor(4);                {get watch cursor for delays}
      snapshot := false;
      for i := 1 to npoints do
         begin
            sampledata[i] := 0;                    {empty incoming data array}
            snapshotdata[i] := 0;                  {empty snapshot data array}
         end;
      whitec.red := $FFFF;
      whitec.blue := $FFFF;
      whitec.green := $FFFF;
      black.red := $0000;
      black.blue := $0000;
      black.green := $0000;
      red.red := $FFFF;
      red.blue := $0000;
      red.green := $0000;
      blue.red := $0000;
      blue.blue := $FFFF;
      blue.green := $0000;
      ltblue.red := $A000;
      ltblue.blue := $FFFF;
      ltblue.green := $E000;
      green.red := $0000;
      green.blue := $0000;
      green.green := $FFFF;                    {define colours}
      options.nwaves := 5;
      options.sinterval := 10;
      options.minvolt := 0;
      options.maxvolt := 5000;                 {define initial parameters}
   end;
```

```
{-------------------------------------------------------------------------------}
{Routine to initialise all of the required MAC OS managers}
{-------------------------------------------------------------------------------}

   procedure initmanagers;
   begin
      InitGraf(@theport);
      InitFonts;
      InitWindows;
      InitMenus;
      TEInit;
      Initdialogs(nil);
      FlushEvents(everyevent, 0);
      InitCursor;
   end;

{-------------------------------------------------------------------------------}
{Routine to setup and draw menus}
{-------------------------------------------------------------------------------}

   procedure setupmenus;
      var
         menubar: handle;
   begin
      menubar := GetNewMBar(MenuBarId);
      if menubar = nil then
         ExitToShell;
      SetMenuBar(menubar);
      DisposeHandle(menubar);
      AddResMenu(GetMHandle(AppleMId), 'DRVR');        {make apple menu}
      DrawMenuBar;
   end;

{-------------------------------------------------------------------------------}
{Handle all events in menu bar ie menu selected}
{-------------------------------------------------------------------------------}

   procedure DoMenuCommand (mresult: longint);
      var
         thememu: integer;
         theitem: integer;
         name: Str255;
         myresult: integer;
   begin
      theitem := loword(mresult);              {get menu item}
      thememu := hiword(mresult);              {get which menu}
      case thememu of
         AppleMId:
            case theitem of
               1:
```

```
                ;
            otherwise
                begin
                    getitem(GetMHandle(AppleMId), theitem, name);
                    myresult := opendeskacc(name);
                end;
        end;                    {handle apple menu}
    FileMId:
        case theitem of
            1:
                doneflag := true;
            otherwise
        end;                    {handle file menu}
    EditMId:
        begin
            if not systemedit(theitem - 1) then
                begin
                end;;
        end;                    {handle edit menu}
    ToolsMId:
        case theitem of
            1:
                doscan;
            3:
                DoPrefs;
            otherwise
        end;                    {our menu, either prefs or scanning}
    otherwise
  end;
  hilitemenu(0);            {actually hilite the menu}
end;

{-------------------------------------------------------------------------------}
{Routine to handle mouse clicks anywhere on screen}
{-------------------------------------------------------------------------------}

procedure domousedown (myevent: eventrecord);     {mouse down event}
  var
    goawayflag: boolean;
    dragrect: rect;
begin
  with screenbits.bounds do
    setrect(dragrect, 4, 24, right - 4, bottom - 4);
      {call quickdraw to set dragging boundaries; ensurethat}
      {at least 4 by 4 pixels will remain visible}
  case findwindow(myevent.where, whichwindow) of
    inSysWindow:
      SystemClick(myevent, whichwindow);   {in other window}
    inmenubar:                  {in menubar}
      DoMenuCommand(menuselect(myevent.where));
```

```
            indrag:                    {titlebar-call window manager to drag}
               dragwindow(whichwindow, myevent.where, dragrect);
            incontent:          {in window content, make window active}
               if whichwindow <> frontwindow then
                  selectwindow(whichWindow);
            ingoaway:           {in close box}
               begin
                  goawayflag := trackgoaway(whichWindow, myevent.where);
                        {track the mouse in the closebox if button pressed}
                  if goawayflag then
                     hidewindow(whichWindow);
               end;
            otherwise
         end;
      end;

{-------------------------------------------------------------------------------}
{Routine to handle keyboard events}
{-------------------------------------------------------------------------------}

      procedure dokeydown (myevent: eventrecord);
         var
            mykey: char;
      begin
         mykey := chr(BitAnd(myevent.message, charCodeMask));
         if BitAnd(myevent.modifiers, cmdkey) <> 0 then
            DoMenuCommand(MenuKey(mykey)); {command key->do menu}
      end;

{-------------------------------------------------------------------------------}
{Routine to handle main event loop of program}
{-------------------------------------------------------------------------------}

      procedure maineventloop;
         var
            myevent: eventrecord;
            theitemhit: integer;
      begin
         repeat
            if getnextevent(everyevent, myevent) then
                     {get next event application should handle}
               begin
                  case myevent.what of       {case on the event}
                     mousedown:       {mousebutton down-call procedure to handle}
                        domousedown(myevent);
                     keydown, autokey:  {keyboard event}
                        dokeydown(myevent);
                     otherwise
                  end;
               end;
```

```
        until doneflag;
    end;

{************}
{Main program}
{************}

begin
    initmanagers;
    setupmenus;
    initthings;
    initcursor;
    maineventloop;
end.
```

3. Logging package – Tattletale program

```
//** Tattletale Model 8 program to simultaneously sample
//** sensor resistance and gas pressures at user selectable
//** sensor temperatures and sampling rates.

//** Compiler directives

      model 800          //** Use model 8
      extension KbHit,KbChar
      extension HPSleep  //** Enable Extra Instructions
      CBreak exit        //** If Ctrl-C Pressed return to Main Menu **

//** Constants and variables

      smin! = 0
      smax! = 5000       //** Sensor heater voltage limits
      atm! = 0
      pressure = 0
      senres! = 0
      senval! = 0        //** ADC inputs and temp storages
      choice = 0         //** User input for action
      samtime = 10
      dacbit! = 0.051
      inc = 0            //** Init Variables and declare constants **
      Heater! = 0
      SDO 255,8          //** Set DAC to Min Volts **
      HPSleep(0)         //** Init sleep

      collect = 0        //** Flag for sampling

//** Main loop of program, detects user input, samples etc **

      print
      input"    Enter Sampling Interval : "samtime
      print
      input"    Enter Heater Voltage : "Heater
      print
      print"    Press g to start ramp! "
      print
      print"Pressure Conductance"          //** Show menu first time
      SDO INT(Heater*dacbit),8           //** Set DAC to User Volts **

start: iff KbHit() > 0
        choice = KbChar()
        iff choice = 81
          goto exit
        endif
        iff choice = 71
```

```
        collect = 1
      endif
      iff choice = 83                    //** Detect user input
        collect = 0
        print
        input"     Enter Sampling Interval : "samtime
        print
        input"     Enter Heater Voltage : "Heater
        print
        print"     Press g to start ramp! "
        print
        print"Pressure Conductance"        //** Show menu again after 1st run
        SDO INT(Heater*dacbit),8           //** Set DAC User Volts **
      endif
    endif
    iff collect = 1                        //** User wants to sample...
      HPSleep(samtime)
      pressure = 0
      senval = 0
      for i = 1 to 10
        pressure = pressure + chan(5)/16
        senval = senval + chan(4)/16
      next i                               //** 10 samples/output to minimise
noise
      senval = senval/10000
      senres = (10000 / (exp(log(10) * ((-senval - 0.006) / 2)))) - 10000
                                           //** Convert ADC value to resistance
      atm = (pressure/20 - 455.)/(4050.-450.) //** Low = Vac Reading High = 1
Atm          Reading
                                           //** Convert ADC value to pressure
      print pressure/20," ",atm," ",senres       //** Print out data
    endif
    goto start

//** User finished so exit program **

exit:  print"Hope you had a nice day....."
```

4. Oxygen adsorption simulation

```
{-------------------------------------------------------------------------------}
{The following ANSI Pascal program uses the Monte Carlo technique to}
{simulate the dissociative adsorption of oxygen onto a surface}
{-------------------------------------------------------------------------------}

Program Simulate;
USES
      Types, QuickDrawText, Events, SegLoad, Fonts, Windows, Devices,
Textutils,
      Menus, Serial, Processes, OSUtils, QuickDraw, TextEdit, Dialogs, Sound;
CONST
   N = 1000;              {No. of sites}
   noavg = 1000;          {No. of MC Steps}
VAR
   qd: QDGlobals;
   site1,site2: integer;                  {sites!}
   surface: array[1..N] of integer;       {surface!}
   i, j:integer;                          {loop counters}
   oxygentheta,avgoxygentheta: longint; {surface coverage}
   O2,kao,kdo:integer;                    {parameters and rate constants}
   OInt,OFin,OStep:integer;               {pressures}
   choice:integer;                        {user choice}

{-----------------------------------------}
{Initialize everything for the program}
{-----------------------------------------}

Procedure Initialize;
begin
   InitGraf(@qd.thePort);
   InitFonts;
   InitWindows;
   InitMenus;
   TEInit;
   InitDialogs(NIL);
   InitCursor;            {Initialize all the needed managers}
   GetDateTime (qd.RandSeed);   {Make Random numbers truly random by
                                 setting seed to actual time (old programmer's
                                 trick!)}
end;
```

```
{-------------------------------------------------------------}
{The following procedures are the various gas/surface}
{interactions that may occur}
{-------------------------------------------------------------}

{* Oxygen adsorption with dissociation *}

Procedure OxygenAds;
begin
   site1 := abs(random mod N) + 1;          {pick a site}
   site2 := abs(random mod N) + 1;          {pick a site}
   if (surface[site1]=0) and (surface[site2]=0) then    {determine if sites vacant}
      begin
         surface[site1] := 1;
         surface[site2] := 1;          {if vacant, occupy them with O's}
      end;
end;

{* Oxygen desorption with association *}

Procedure OxygenDes;
begin
   site1 := abs(random mod N) + 1;          {pick a site}
   site2 := abs(random mod N) + 1;          {pick a site}
   if (surface[site1]=1) and (surface[site2]=1) then    {determine if both sites
                                                         occupied}
      begin
         surface[site1] := 0;
         surface[site2] := 0;          {if occupied, vacate them}
      end;
end;

{-------------------------------------------------------}
{The following procedure is the actual simulation}
{-------------------------------------------------------}

Procedure O2Simulation;
   var
      i,j:integer;
begin
   write('Enter Oxygen kao Value: ');
   readln(kao);
   write('Enter Oxygen kdo Value: ');
   readln(kdo);
   write('Enter Initial Oxygen Pressure: ');
   readln(OInt);
   write('Enter Final Oxygen Pressure: ');
   readln(OFin);
   write('Enter Oxygen Pressure Step Size: ');
   readln(OStep);    {get user inputs}
```

```
    writeln(' O2 ',' oxygentheta ');
    O2 := OInt;
    repeat
      avgoxygentheta := 0;
      for i := 1 to noavg do        {do it many times so equilibrium attained}
        begin
          for j := 1 to (kao * O2) do
            OxygenAds;              {adsorption}
          for j := 1 to kdo do
            OxygenDes;              {desorption}
          oxygentheta := 0;
          for j := 1 to N do
            if (surface[j]=1) then
              oxygentheta := oxygentheta + 1;
              {measure oxygensurface coverage}
          avgoxygentheta := avgoxygentheta + oxygentheta;    {add to total}
        end;
      writeln((avgoxygentheta/noavg)/N:6:4); {calculate and output results}
      O2 := O2 + OStep;
    until O2 > OFin
end;

{-----------------}
{* Main body *}
{-----------------}

BEGIN
    Initialize;
    choice := 0;
    for i := 1 to N do
      surface[i] := 0;              {make all sites unoccupied}
    repeat
      writeln;
      writeln('Menu...');
      writeln;
      writeln('(1. Langmuir (Oxygen) Modelling)');
      writeln('(2. Quit');
      writeln;
      write('Choice : ');
      readln(Choice);               {generate menu options and user input}
      if Choice = 1 then            {simulate it...}
        O2Simulation;
    until choice = 2;
end.
```

5. Hinshelwood simulation – single analyte

```
{-------------------------------------------------------------------------------}
{The following ANSI Pascal program uses the Monte Carlo technique to}
{simulate the dissociative adsorption of oxygen onto a surface}
{with the competitive non-dissocative adsorption of an analyte,}
{and the surface reactions between the adsorbed oxygen and analyte}
{-------------------------------------------------------------------------------}

Program Simulate;
USES
      Types, QuickDrawText, Events, SegLoad, Fonts, Windows, Devices,
Textutils,
      Menus, Serial, Processes, OSUtils, QuickDraw, TextEdit, Dialogs, Sound;
CONST
   N = 1000;              {No. of sites}
   noavg = 1000;          {No. of MC Steps}
VAR
   qd: QDGlobals;
   site1,site2: integer;                {sites!}
   surface: array[1..N] of integer;     {surface!}
   i, j:integer;                        {loop counters}
   oxygentheta,avgoxygentheta: longint; {surface coverage}
   O2,PA,kao,kdo,kaa,kda,kra:integer;   {parameters and rate constants}
   OInt,OFin,AInt,AFin,OStep,AStep:integer;{pressures}
   choice:integer;                      {user choice}

{-----------------------------------------}
{Initialize everything for the program}
{-----------------------------------------}

Procedure Initialize;
begin
   InitGraf(@qd.thePort);
   InitFonts;
   InitWindows;
   InitMenus;
   TEInit;
   InitDialogs(NIL);
   InitCursor;            {Initialize all the needed managers}
   GetDateTime (qd.RandSeed);   {Make Random numbers truly random by
                                 setting seed to actual time (old programmer's
                                 trick!)}
end;
```

```
{-------------------------------------------------------------}
{The following procedures are the various gas/surface}
{interactions that may occur}
{-------------------------------------------------------------}

{* Oxygen adsorption with dissociation *}

Procedure OxygenAds;
begin
   site1 := abs(random mod N) + 1;          {pick a site}
   site2 := abs(random mod N) + 1;          {pick a site}
   if (surface[site1]=0) and (surface[site2]=0) then    {determine if sites vacant}
      begin
         surface[site1] := 1;
         surface[site2] := 1;          {if vacant, occupy them with O's}
      end;
end;

{* Oxygen desorption with association *}

Procedure OxygenDes;
begin
   site1 := abs(random mod N) + 1;                  {pick a site}
   site2 := abs(random mod N) + 1;                  {pick a site}
   if (surface[site1]=1) and (surface[site2]=1) then       {determine if both sites
                                                            occupied}
      begin
         surface[site1] := 0;
         surface[site2] := 0;          {if occupied, vacate them}
      end;
end;

{* Analyte A adsorption *}

Procedure AnalyteAAds;
begin
   site1 := abs(random mod N) + 1;          {pick a site}
   if (surface[site1]=0) then       {determine if site vacant}
      begin
         surface[site1]:=2;         {if vacant, occupy it}
      end;
end;

{* Analyte A desorption *}

Procedure AnalyteADes;
begin
   site1 := abs(random mod N) + 1;          {pick a site}
   if (surface[site1]=2) then.      {determine if site occupied}
      begin
```

```
            surface[site1]:=0;          {if occupied, vacate it}
        end;
end;

{* Surface reaction between an adsorbed analyte(A) and an adsorbed oxygen *}

Procedure AandO;
begin
    site1 := abs(random mod N) + 1;             {pick a site}
    site2 := abs(random mod N) + 1;             {pick a site}
    if ((surface[site1]=1) and (surface[site2]=2)) or ((surface[site1]=2) and
    (surface[site2]=1)) then                    {determine if sites are occupied}
        begin
            surface[site1] := 0;
            surface[site2] := 0;        {if occupied, vacate them}
        end;
end;

{-------------------------------------------------------}
{The following procedure is the actual simulation}
{-------------------------------------------------------}

Procedure HinshelwoodAandO;
    var
        i,j:integer;
begin
    write('Enter Oxygen Pressure: ');
    readln(O2);
    write('Enter Oxygen ka Value: ');
    readln(kao);
    write('Enter Oxygen kd Value: ');
    readln(kdo);
    write('Enter Analyte ka Value: ');
    readln(kaa);
    write('Enter Analyte kd Value: ');
    readln(kda);
    write('Enter Reaction kr Value: ');
    readln(kra);
    write('Enter Initial Analyte A Pressure: ');
    readln(AInt);
    write('Enter Final Analyte A Pressure: ');
    readln(AFin);
    write('Enter Analyte A Pressure Step Size: ');
    readln(AStep);
    writeln;                    {get user inputs}
    writeln(' Oxygentheta ');
    PA := AInt;
    repeat
        avgoxygentheta := 0;
        for i := 1 to noavg do      {do it many times so equilibrium attained}
```

```
        begin
          for j := 1 to (kao * O2) do
            OxygenAds;              {oxygen adsorption}
          for j := 1 to kdo do
            OxygenDes;              {oxygen desorption}
          for j := 1 to (kaa * PA) do
            AnalyteAAds;            {analyte adsorption}
          for j := 1 to kda do
            AnalyteADes;            {analyte desorption}
          for j := 1 to kra do
            AandO;                  {Hinshelwood reaction}
          oxygentheta := 0;
          for j := 1 to N do
            if (surface[j]=1) then
              oxygentheta := oxygentheta + 1;
              {measure oxygensurface coverage}
          avgoxygentheta := avgoxygentheta + oxygentheta;    {add to total}
        end;
      writeln((avgoxygentheta/noavg)/N:6:3);  {calculate and output results}
      PA := PA + AStep;
  until PA > AFin;
  writeln;
end;

{-----------------}
{* Main body *}
{-----------------}

BEGIN
  Initialize;
  choice := 0;
  for i := 1 to N do
    surface[i] := 0;                {make all sites unoccupied}
  repeat
    writeln;
    writeln('Menu...');
    writeln;
    writeln('(1. Hinshelwood (Two Analyte) Modelling)');
    writeln('(2. Quit)');
    writeln;
    write('Choice : ');
    readln(Choice);                 {generate menu options and user input}
    writeln;
    if Choice = 1 then              {simulate it...}
      HinshelwoodAandO;
  until choice = 2;
end.
```

6. Hinshelwood simulation – two analytes

```
{-------------------------------------------------------------------------------}
{The following ANSI Pascal program uses the Monte Carlo technique to}
{simulate the dissociative adsorption of oxygen onto a surface}
{with the competitive non-dissocative adsorption of two analytes,}
{and the surface reactions between the adsorbed oxygen and analytes}
{-------------------------------------------------------------------------------}

Program Simulate;
USES
      Types, QuickDrawText, Events, SegLoad, Fonts, Windows, Devices,
Textutils,
      Menus, Serial, Processes, OSUtils, QuickDraw, TextEdit, Dialogs, Sound;
CONST
   N = 1000;              {No. of sites}
   noavg = 1000;          {No. of MC Steps}
VAR
   qd: QDGlobals;
   site1,site2: integer;                {sites!}
   surface: array[1..N] of integer;     {surface!}
   i, j:integer;                        {loop counters}
   oxygentheta,avgoxygentheta: longint;              {surface coverage}
   O2,PA,PB,kao,kdo,kaa,kda,kab,kdb,kra,krb:integer;     {parameters and
                                                        rate constants}
   OInt,OFin,AInt,AFin,BInt,BFin,OStep,AStep,BStep:integer;{pressures}
   choice:integer;                                   {user choice}

{-----------------------------------------}
{Initialize everything for the program}
{-----------------------------------------}

Procedure Initialize;
begin
   InitGraf(@qd.thePort);
   InitFonts;
   InitWindows;
   InitMenus;
   TEInit;
   InitDialogs(NIL);
   InitCursor;              {Initialize all the needed managers}
   GetDateTime (qd.RandSeed);   {Make Random numbers truly random by
                                Setting Seed to actual time (old programmers
                                technique)}
end;
```

```
{-------------------------------------------------------------}
{The following procedures are the various gas/surface}
{interactions that may occur}
{-------------------------------------------------------------}

{* Oxygen adsorption with dissociation *}

Procedure OxygenAds;
begin
   site1 := abs(random mod N) + 1;          {pick a site}
   site2 := abs(random mod N) + 1;          {pick a site}
   if (surface[site1]=0) and (surface[site2]=0) then    {determine if sites vacant}
      begin
         surface[site1] := 1;
         surface[site2] := 1;          {if vacant, occupy them with O's}
      end;
end;

{* Oxygen desorption with association *}

Procedure OxygenDes;
begin
   site1 := abs(random mod N) + 1;                {pick a site}
   site2 := abs(random mod N) + 1;                {pick a site}
   if (surface[site1]=1) and (surface[site2]=1) then      {determine if both sites
                                                          occupied}
      begin
         surface[site1] := 0;
         surface[site2] := 0;          {if occupied, vacate them}
      end;
end;

{* Analyte A adsorption *}

Procedure AnalyteAAds;
begin
   site1 := abs(random mod N) + 1;          {pick a site}
   if (surface[site1]=0) then       {determine if site vacant}
      begin
         surface[site1]:=2;         {if vacant, occupy it}
      end;
end;
```

```
{* Analyte A desorption *}

Procedure AnalyteADes;
begin
   site1 := abs(random mod N) + 1;          {pick a site}
   if (surface[site1]=2) then      {determine if site occupied}
      begin
         surface[site1]:=0;        {if occupied, vacate it}
      end;
end;

{* Analyte B adsorption *}

Procedure AnalyteBAds;
begin
   site1 := abs(random mod N) + 1;          {pick a site}
   if (surface[site1]=0) then      {determine if site vacant}
      begin
         surface[site1]:=3;        {if vacant, occupy it}
      end;
end;

{* Analyte B desorption *}

Procedure AnalyteBDes;
begin
   site1 := abs(random mod N) + 1;          {pick a site}
   if (surface[site1]=3) then      {determine if site occupied}
      begin
         surface[site1]:=0;        {if occupied, vacate it}
      end;
end;

{* Surface reaction between an adsorbed analyte(A) and an adsorbed oxygen *}

Procedure AandO;
begin
   site1 := abs(random mod N) + 1;          {pick a site}
   site2 := abs(random mod N) + 1;          {pick a site}
   if ((surface[site1]=1) and (surface[site2]=2)) or ((surface[site1]=2) and
   (surface[site2]=1)) then                {determine if sites are occupied}
      begin
         surface[site1] := 0;
         surface[site2] := 0;      {if occupied, vacate them}
      end;
end;

{* Surface reaction between an adsorbed analyte(B) and an adsorbed oxygen *}

Procedure BandO;
```

```
begin
   site1 := abs(random mod N) + 1;          {pick a site}
   site2 := abs(random mod N) + 1;          {pick a site}
   if ((surface[site1]=1) and (surface[site2]=3)) or ((surface[site1]=3) and
   (surface[site2]=1)) then                 {determine if sites are occupied}
      begin
         surface[site1] := 0;
         surface[site2] := 0;      {if occupied, vacate them}
      end;
end;

{-------------------------------------------------------}
{The following procedure is the actual simulation}
{-------------------------------------------------------}

Procedure HinshelwoodABandO;
   var
      i,j:integer;
begin
   write('Enter Oxygen Pressure: ');
   readln(O2);
   write('Enter Oxygen ka Value: ');
   readln(kao);
   write('Enter Oxygen kd Value: ');
   readln(kdo);
   write('Enter Analyte A ka Value: ');
   readln(kaa);
   write('Enter Analyte A kd Value: ');
   readln(kda);
   write('Enter Reaction A kr Value: ');
   readln(kra);
   write('Enter Analyte B ka Value: ');
   readln(kab);
   write('Enter Analyte B kd Value: ');
   readln(kdb);
   write('Enter Reaction B kr Value: ');
   readln(krb);
   write('Enter Initial Analyte A Pressure: ');
   readln(AInt);
   write('Enter Final Analyte A Pressure: ');
   readln(AFin);
   write('Enter Analyte A Pressure Step Size: ');
   readln(AStep);
   write('Enter Initial Analyte B Pressure: ');
   readln(BInt);
   write('Enter Final Analyte B Pressure: ');
   readln(BFin);
   write('Enter Analyte B Pressure Step Size: ');
   readln(BStep);
   writeln;                  {get user inputs}
```

```
    writeln(' Oxygentheta ');
    PA := AInt;
    repeat
      PB := BInt;
      repeat
        avgoxygentheta := 0;
        for i := 1 to noavg do        {do it many times so equilibrium attained}
          begin
            for j := 1 to (kao * O2) do
              OxygenAds;              {oxygen adsorption}
            for j := 1 to kdo do
              OxygenDes;              {oxygen adsorption}
            for j := 1 to (kaa * PA) do
              AnalyteAAds;            {analyte A adsorption}
            for j := 1 to kda do
              AnalyteADes;            {analyte A desorption}
            for j := 1 to (kab * PB) do
              AnalyteBAds;            {analyte B adsorption}
            for j := 1 to kdb do
              AnalyteBDes;            {analyte B desorption}
            for j := 1 to kra do
              AandO;                  {Hinshelwood reaction O and A}
            for j := 1 to krb do
              BandO;                  {Hinshelwood reaction O and B}
            oxygentheta := 0;
            for j := 1 to N do
              if (surface[j]=1) then
                oxygentheta := oxygentheta + 1;    {measure oxygensurface
                                                    coverage}
            avgoxygentheta := avgoxygentheta + oxygentheta;   {add to total}
          end;
        writeln((avgoxygentheta/noavg)/N:6:3); {calculate and output results}
        PB := PB + BStep;
      until PB > BFin;
      PA := PA + AStep;
    until PA > AFin;
    writeln;
end;

{------------}
{* Main body *}
{------------}

BEGIN
    Initialize;
    choice := 0;
    for i := 1 to N do
      surface[i] := 0;                {make all sites unoccupied}
    repeat
      writeln;
```

```
        writeln('Menu...');
        writeln;
        writeln('(1. Hinshelwood (Two Analyte) Modelling)');
        writeln('(2. Quit)');
        writeln;
        write('Choice : ');
        readln(Choice);                    {generate menu options and user input}
        writeln;
        if Choice = 1 then                 {simulate it...}
           HinshelwoodABandO;
    until choice = 2;
end.
```